

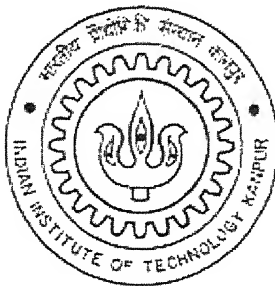
CONSTRUCTION OF SOLID MODEL FROM ORTHOGRAPHIC VIEWS - A KNOWLEDGE BASED APPROACH

A Thesis submitted
in partial fulfillment of the requirements
for the degree of

Master of Technology

by

KORLEPARA N S SUDHEER



to the

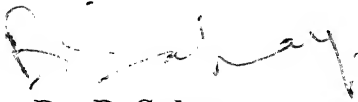
DEPARTMENT OF MECHANICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

July 2001

34-7-2001
R

CERTIFICATE

It is certified that the work contained in the thesis entitled **Construction of Solid model from orthographic views-A knowledge based approach**, by **Korlepara N S Sudheer**, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.



Dr. B. Sahay

Professor

Department of Mechanical Engineering
Indian Institute of Technology Kanpur

July, 2001

9 FEB 2003 /ME

महाराष्ट्र शासन केलकर पुस्तकालय

सामाजिक शोधोपयोगी संस्थान कानपुर

अवाप्ति क्र० A-141967



A141967

ACKNOWLEDGEMENTS

During the course of my M.Tech Thesis, many people have helped me, in various ways. Each one has played a role in helping me during my stay here. My gratitude to all of them knows no bounds.

First of all I would like to thank almighty, with out his help I know I could have done nothing. The way the things turned to their worst best towards the end, I think only by gods grace I have completed this work.

I would like to thank my parents and my sister for their constant encouragement. During my stay (to be correct, overstay) here, those phone calls certainly made me to work faster in order to go home to meet them.

I am very much grateful to my thesis supervisor, Prof. B. Sahay, for his valuable guidance rendered to me during the course of the present work. Engineering Drawing is my favorite subject right from my first year of graduation. I am thankful to him for giving me this topic as my M.Tech thesis. Thinking the magnitude of the problems I have created towards the end, I know any other guide would have shown me the exit door. I can say with out his help in so many crunch situations I don't think I would have successfully completed my degree. His patience, his forgiveness, his helping nature are the things I am going to remember throughout my life. Once again I thank him for helping me to get out of many administrative hurdles.

I gratefully acknowledge Prof. Amitabha Mukharjee, for the CAD course he offered. After doing that course only I have oriented myself towards CAD. The way he dealt that course helped me a lot to gain the concepts of solid modeling, computational geometry and of course data structures. Incidentally he was my first guide.

Shankar, what can I say about him. All of those 25 days I spent at HC, he came there. He looked after me to the maximum extent one can. I don't want to remember those horrifying days, but I certainly remember them as the time he has showed how good friend he is. I can only say he has given me new life.

Parthasarathy has provided the moral support with out which I would have definitely seen the limits of frustration because of my overstay. We both came here together, stayed as roommates for 2 years. Luckily for me, he also stayed here up to the

time I have completed my work. He not only helped me whenever he was free but also when he was neck deep with his own work. I thank him very much once again.

Thanks for tanga murali for reminding many times that the time is running out.

Among juniors, I thank all of them, who have helped me to complete my work. Laxman, Suresh Behara, Yugandar, Dali Raju, Raja have helped me towards the end a lot. Keeping in view the amount of time left I don't think I would have completed my work before due date with out their help. Sampath was like a godsend. The help he has done in the last two days was instrumental in defending my thesis on 27 July.

Finally I would like to thank Dr. V.Nigam, Dr. D.K.Sinha, Dr. Raka Kaushal, and Dr. Amit Gupta for treating me to get cured of my disease.

Korlepara N. S. Sudheer

Dedicated to

My parents
and
my beloved sister

ABSTRACT

Computer- based system for modeling the geometry of rigid solid objects are becoming increasingly important in various applications of mechanical, civil engineering, computer graphics, computer vision and other fields. At the heart of such systems is symbolic representation of *abstract solids* that model the real world objects. However most of the industries use the way of orthographic projections because of the ease they provide in creating and understanding them. Also it is very easy for humans to share the information between each other.

Present work deals with reconstruction of solids from engineering drawings. The algorithm presented here is robust and solves the geometries involving polyhedral and cylindrical objects. This work can be thought as a direction in the way of developing a generalized reconstruction algorithm to solve any type of problem.

The algorithm is a knowledge based search algorithm. The main underlying philosophy is to view a complex part as being composed of elementary objects such as arcs, triangles, and quadrilaterals and to recognize these elementary objects by making use of the knowledge about the class dependent patterns of their 2D representations. This approach has the potential for dealing with a variety of non-polyhedral objects.

This involves 4 steps: preprocessing of input data to generating all possible loops by heuristic search; discretization of these loops to form basic primitives such as arcs, triangles, quadrilaterals; map these primitives to form definite 3D primitives; to decide whether the primitive generated is a void or solid.

This algorithm is tested in case of many practical drawings and correct results are obtained. This work can be further extended to include spherical, conical surfaces.

Contents

Title	i
Certificate	ii
Acknowledgements	iii
Dedication	v
Abstract	vi
List of figures	xi
1. Introduction	1
1.1 Literature Review	2
1.1.1 Wireframe oriented Approach	3
1.1.2 Volume oriented approach	4
1.2 Objective of present work	6
1.3 Organization of the Thesis	7
2. Representation of Rigid solids	8
2.1 Representation scheme	8
2.2 Schemes for representing Rigid Solids	10
2.2.1 Constructive Solid Geometry (CSG)	10
CSG Trees	10
Properties of CSG Schemes	10
2.2.2. Sweep Representation	12
Translational Sweep	12
Rotational Sweep	13
2.2.3 Engineering Drawings	13

3. Knowledge of Engineering Drawing	14
3.1 Importance of engineering drawings	14
3.2 Elements of engineering drawings	14
3.2.1 Geometry information of the object	15
a) Orthographic Projections	16
More than one orthographic view is necessary	18
Types of orthographic projections	18
i. First angle projections	18
ii. Third angle projection	19
b) Auxiliary projections	20
3.2.2 Understanding the Geometry information	21
3.2.3 Understanding other information	21
Symbols	21
Annotations	22
Dimensioning	22
Data Base	23
3.2.4 understanding engineering drawing by computer	23
4. The Algorithms	25
4.1 Knowledge of 2D representation for 3D objects	25
4.2 The input	26
4.3 The loop	27
4.4 Extraction of the loops from orthographic projections	28
4.4.1 Tree structure	28

4.4.2 Building Tree	28
4.4.3 Obtaining the loops from the tree	30
4.4.4 Making a loop anticlockwise	30
4.4.5 Deleting the repetitive loops	31
4.4.6 Redundant Nodes	32
4.5 The Discretization	32
4.5.1 The termination criteria	32
4.5.2 Separation of arcs	33
4.5.3 Separation of triangular primitives	34
4.5.4 Separation by ray casting	35
4.5.5 Separating inclined lines	37
4.6 Constructing 3D primitives from 2D primitives	41
Finding the coordinates of a 3D primitive	45
4.7 Special case of an arc being tangent	46
4.8 Classification of 3D primitives	48
4.8.1 Sign associated with 2D primitive	48
4.8.2 Hidden line information	48
4.8.3 Position in space	49
Volume enclosure relationships	49
Rules to classify subparts	49
Intersecting subparts	50
4.9 Boolean operations	51

5. Results	52
6. Conclusions and Scope of Future Work	64
6.1 Achievements	64
6.2 Limitations	65
6.3 Scope for future work	66
Bibliography	68
Appendix	71
Input format	71
Notations used	71
Points connected by a line segment	72
Points connected by an arc segment	73

List of figures

2.1	Domain and range of a representation scheme	8
2.2	Classification of representation schemes	9
2.3	A CSG Tree and solids represented by its subtree	11
2.4	Two CSG schemes having different primitives but same domain	11
2.5	Translational sweep	12
2.6	Rotational sweep	13
3.1	A typical Engineering drawing	15
3.2	The glass box	17
3.3	The glass box opened out	17
3.4	The principal planes of projection	19
3.5	First angle and Third angle projections	19
3.6	Auxiliary view of a wedge block	20
4.1	knowledge of 2D representation for rectangular prism and cylinder	25
4.2	A Typical orthographic view	26
4.3	Loops formed from the orthographic view	27
4.4	A Tree structure	28
4.5	Tree structures formed from the orthographic view in figure 4.2	29
4.6	A clockwise connected loop	31
4.7	Loops with CW and ACW arcs	33
4.8	Loops shown in figure 4.7 after arc separation	33
4.9	Loop having separable triangular primitive	34

4.10	Loop in figure 4.9 after triangular separation	35
4.11	Separation by casting a ray	36
4.12	Inclined line separation	37
4.13	Acceptable cases for forming a positive triangular primitive	38
4.14	Acceptable cases for forming a positive trinangular primitive	38
4.15	Acceptable cases for forming a negative trinangular primitive	40
4.16	Orthographic views of a quadrilateral in various positions	42
4.17	Orthographic views of a triangle in various positions	43
4.18	Orthographic views of an arc in various positions	44
4.19	Coordinate system	45
4.20	Position of the x-y-z axes	45
4.21	Case of an arc being tangential to an edge	46
4.22	Line segments introduced in figure 4.21	47
4.23	Case of one subpart is IN in two other subparts	49
4.24	View of a typical orthographic drawing	50
4.25	Subparts after reconstruction for figure 4.24	50
5.1	Reconstruction of a rectangular prism	54
5.2	Reconstruction of a triangular prism	55
5.3	Reconstruction of a cylinder	56
5.4	Reconstruction of an object having rectangular, triangular primitives	57
5.5	Reconstruction of an object having rectangular primitives	58
5.6	Reconstruction of an object having rectangular, circular primitives case (i)	59
5.7	Reconstruction of an object having rectangular, circular primitives case (ii)	60

5.8	Reconstruction of an object having rectangular, triangular, cylindrical primitives case- (i)	61
5.9	Reconstruction of an object having rectangular, triangular, cylindrical primitives case-(ii)	62
5.10	Reconstruction of an object having rectangular, triangular, cylindrical primitives case-(iii)	63
5.11	Reconstruction of an object having rectangular, triangular, cylindrical primitives case-(iv)	64
A-1	nodes connected by line segments	72
A-2	nodes connected by an arc segment	73
A-3	a typical orthographic view	74

Chapter 1

Introduction

In the process of computer aided mechanical engineering design and manufacturing, representation of solid model in computer is essential. This type of representation can be utilized in applications such as mass property calculations, mechanism analysis, finite element modeling and NC machining etc.

The approaches generally used for representation of solids are

- Pure primitive Instancing
- Spatial occupancy enumeration
- Cell decomposition
- Constructive solid geometry
- Sweep representations
- Boundary representation
- Engineering drawings.

It is common engineering practice to generate two or more two dimensional orthographic views to convey the form of three-dimensional object. Most existing products are presented by means of engineering drawings with blue prints or 2D CAD systems. This became traditional means for specifying solids, as they are best for informal means of communication between humans.

An engineer is trained to interpret these projections and can create a solid model out of them. This assumes certain knowledge has been imparted to him in the form of geometry and conventions used. Using his knowledge, from any complicated blue print drawing in its two dimensions, he can visualize the same object in its three dimensions along with the machining processes used, tolerance limits of the various components, type of the material, list of the parts, quantity of each part required, etc. If a detailed drawing of a machine part is provided to him, he needs no other information to manufacture the same.

With the advent of computer and the subsequent revolution it brought in to the field of engineering, industry has demanded automated computerized systems that can handle all the required design processes. First step in any design process is to identify a blue print drawing. Various researchers have tried to impart the knowledge of interpreting these drawings to computers. To automatically interpret these drawings, computers has to understand the conventions and standards used in them. The annotation and the other layers of information have to be removed. Finally, using the geometric projections of the shape in two dimensions, a three dimensional object has to be created.

This process of reconstruction is known as *solid model reconstruction from orthographic projections*. Although obtaining 2D projections from a given 3D object is very straight forward, the reverse process is quite difficult. The difficulties of this approach arise from the loss of semantics information when a 3D object is represented in 2D. Ample work has been done in this way but nothing can be told as robust. So all the commercial packages like I-DEAS, NASTRAN etc use other approaches of solid modeling.

1.1 Literature Review

Based on the apparent differences used in the reconstruction process, these are classified in to two types. They are

- *bottom-up* approach
- *top-down* approach

A *bottom-up* approach is one, which starts from 2D entities and works its way up to form a 3D object. On the other hand *top-down* approach starts from constructing a 3D object first and operates on it until its three views match the given drawing.

To date researchers have come up with an array of algorithms, which deal with the generation of solid models from orthographic views. Idesawa [1,2] published first paper in this field. Several algorithms have been proposed after that, but none of them can solve the problem completely. These algorithms can be divided into two types:

- wireframe oriented
- volume oriented

Wireframe approach is a *bottom-up* one. It starts with 2D nodes and results in a 3D wireframe. The real object is found by propagating this wireframe. Much research has been done on this approach in recent years.

Volume approach involves forming primitives. Then regularized boolean operations [27] are performed on these primitives to identify final object. It can be *bottom-up* or *top-down*.

1.1.1 Wireframe oriented Approach

This is a *bottom-up* approach proposed by Idesawa [1,2]. It consist of following steps:

- Generating candidate 3D vertices from 2D nodes.
- Constructing 3D edges from 3D vertices.
- Forming the face loops of the object from 3D edges.
- Building component of the object from the face loops.
- Combining components to find solution.

This method is generally called *fleshing out projections*. He gave a set of rules for identifying edges and vertices in the process of reconstruction. Although his algorithm can with rectilinear objects, other objects with pathological cases or multiple solution cases cannot be dealt. This method became basis for algorithms developed thereafter.

Shapira [3] used the same approach proposed by Idesawa to solve some of the problems when only two projections are available.

Wesley and Markowasky [4-5] provided a very thorough analysis of both wireframe generation and wireframe to boundary representation (B-rep) in reconstruction of arbitrary polyhedral solids. It uses detailed views and two or more orthographic views to construct a more precise model. It provides good results in multiple-solutions case and pathological cases but is computationally expensive because it performs a large number of re-projections and complicated geometric operations. It is also limited only to polyhedral objects.

As an extension to Wesley's method [4-5], Sakurai [6] proposed an algorithm for reconstruction of cylindrical, conical, toroidal and spherical surfaces that are parallel to

one of the coordinate axes by using vertex type classification method. Because it is based on Wesley's method it is also complicated and computationally expensive.

Yan *et. al.* [7] presented an efficient method for reconstruction of polyhedral objects from orthographic projections based on *bottom-up* approach. Algorithm can handle pathological cases and multiple-solution cases. This work formalizes the line constraints idea by defining the qualitative partitioning of search space. To improve the processing speed of 3D edge generation, the frontal projection based decision tree search technique is used. This work though is several years old is one of the best-known algorithms for reconstruction of rectilinear solids.

Mukerjee *et. al.* [8] reconsidered the problem of converting a linear drawing into a set of 3D edges based on qualitatively complete classification of all possible 3D edges and their projections. The algorithm presented, has used line constraint based approach while most of the previous algorithm have considered either point constraint or a subset of line constraint approach, which showed increase in efficiency. The result presented by them is limited to only polyhedral class of objects. For a complete description of the whole problem reader is directed to read [9-15].

Presently there is no available software, which can implement all the stages of reconstruction process and can be said to be professional. Dori and Tombre [16,17] discussed difficulties in this field and reasons for lack of serious attempt in moving to higher level. Based on their experience, they presented an analysis of mechanical engineering drawing, and a schema is proposed to develop complete software for achieving high-level conversion of technical document of this type. Harlick and Shaprio [18] also discussed various aspects of this problem and provided the merits and demerits of the key work done in this field and finally proposed research directives to enhance the practicality of converting a paper engineering drawing to 3D CAD model.

1.1.2 Volume oriented approach

This approach involves obtaining a solid model directly from orthographic views contrary to previous approach in which an intermediate wire frame comes. Typically the approaches proposed involve translation sweeps and regularized boolean operations.

Aldefeld [19,20] was first to propose this algorithm. He viewed complex part as composed of several elementary objects (primitives). These basic primitives were recognized using heuristic search from specific pattern in the 2D representation. However this algorithm can only reconstruct object of uniform thickness. Input for this algorithm needs to specify each elementary object manually and they are treated as isolated bodies.

Bin [21] also proposed a similar algorithm but with some modifications, his program requires less user interaction and treats a wider range of engineering objects.

Chen [22] proposed an algorithm, which consist of three phases: decomposition, reconstruction and composition. This algorithm is also user interactive and can handle polyhedral objects with non-uniform thickness.

Meeran *et. al.* [23] also proposed a similar algorithm which users sweep operations to generate basic objects and final object is obtained by combining these with boolean operations. His method deals only with uniform thickness solids.

Tanaka *et. al.* [24] presented a method of automatically decomposing a 2D assembly drawing into 3D part drawing using a set of solid element equation. Every solid element is classified in one of three types, a true element, a false element and an undecided element. Result produced by this method was able to generate all possible solutions if more than one solution exists. It can deal with only polyhedral class and cylindrical surfaces parallel to one of the axis.

Shum *et. al.* [25] proposed a two-stage extrusion procedure. In each stage, geometric entities from only three orthogonal views (viz. top, side front) are used. In first stage an exterior contour region in each view is swept along its normal direction according to the corresponding object dimension. As a result three extrusion solids are produced. Intersection of these three basic solids forms a basic solid. Next, all the interior entities of each view are treated by filtering process. This method has limitation because it uses linear extrusion. CSG primitive solids such as sphere, cone, torus or tetrahedron cannot be generated from linear extrusion.

Tomiyama [26] presented a *top-down* approach, which takes the outermost base of all views. Sweeping them along the direction orthogonal to view planes creates 3D initial object. Now comparisons are made between original views and projected view of solid formed to find missing lines, surplus lines, etc.

1.2 Objective of present work

The work presented here aims at reconstructing 3D solid model using volume oriented approach. It uses three orthographic projections namely front, top and side views. Work done in this field of reconstruction generally use wire frame oriented approach. Algorithms proposed in volume oriented approach are very limited and mostly deal with polyhedral objects. Present work reconstructs solid model of projections containing polyhedral and cylindrical objects.

First part of the work involves preprocessing the input data to form a tree structure [28]. A node of a tree branch can have any number of sons but it should have only one father node. Starting from any node of a projection, whole of it is traversed to form closed loops. Whenever a loop is formed branch of the tree is ended. Thus, the number of branches in the tree equals the number of the closed loops in the drawing. Once whole drawing is traversed loops are extracted from the tree by traversing its branches.

Second part of the work involves processing these closed loops to form quadrilateral, cylindrical, and triangular primitives by discretization. A closed loop may have a complex shape having inclined lines and arcs. The objective of this step is to separate these arcs and inclined lines from each loop. The process of separating these shapes from each loop is repeated until it results only in quadrilaterals, triangles, and cylindrical primitives.

Third part of the work deals with all the discretized loops of front, top and side views. Comparisons are made in between these loops to form a set of 3D primitives, which form the solid model. The knowledge of 2D representations and their corresponding 3D form is utilized in deciding the shape of a 3D primitive along with its orientation in three dimensional space.

Last part of the work involves first deciding whether a particular 3D primitive is a solid or void. Comparing the relative position of each primitive in three dimensional space, status of every block is decided. Later this information about these 3D primitives is given to I-DEAS solid modeling software. Using I-DEAS regularized boolean operations are performed and final solid in its three dimensions is realized.

1.3 Organization of the Thesis

This thesis is presented in six chapters.

Chapter one contains introduction to the problem and the literature surveys.

Chapter two gives a review to some of the ways of representing solids. In this chapter engineering drawings, constructive solid geometry representations and sweep representations are discussed.

Chapter three gives the detailed description of engineering drawings.

Chapter four presents the algorithms used in the present work.

Chapter five gives the results.

Chapter six gives the possible future extensions for the present work.

Chapter 2

Representation of Rigid solids

Three dimensional solid geometry plays an important role in many scientific and engineering fields and is vital to industries. The need for powerful computational means for dealing with geometry is widely recognized, and a variety of geometric programs and systems have been or being developed by research laboratories, industry users and commercial vendors.

Geometric algorithms do not manipulate physical solids; rather they manipulate data, which represent solids. The way we take the data from a solid forms its representation. A representation should have syntax (valid notations) and semantics (geometric meaning).

2.1 Representation scheme

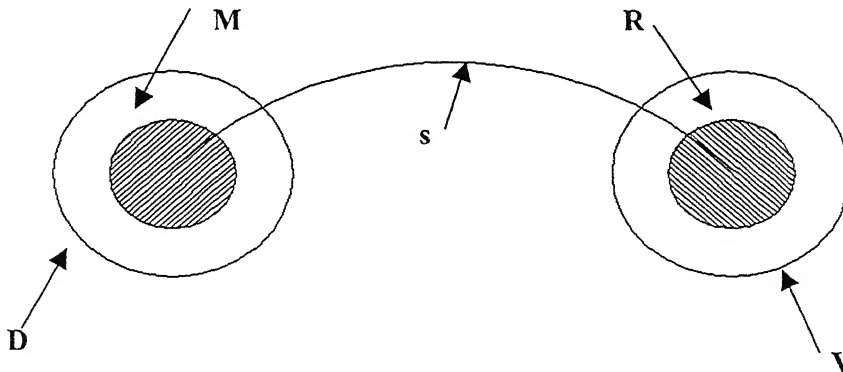


Fig. 2.1: Domain and range of a representation scheme

where:

s : Representation Scheme $M \rightarrow R$.

M : Modeling space

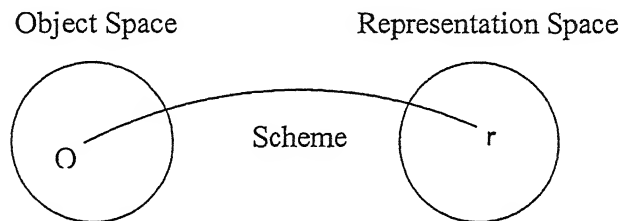
R : Space of syntactically correct representations produced by programmer.

D (domain of s): Set of elements of M that may be represented via s .

V (range of s): Set of syntactically correct representations that are images of D .

A representation scheme is defined as a relation that maps a valid point set in to a valid solid model. Any representation shown in the figure 2.1 in the range V is said to be valid since it is both syntactically valid and semantically correct i.e. it belongs to R and there exists a corresponding element in D .

A scheme is unambiguous or complete, but not unique if more that one model can represent more that one object. A scheme is ambiguous or incomplete if one model can represent more that one object, as in the case of a wireframe model. These schemes are shown in figure 2.2.



a. Unambiguous, complete, unique scheme



b. Unambiguous, complete, non unique scheme



c. Ambiguous, incomplete scheme

Fig. 2.2: Classification of representation schemes

2.2 Schemes for representing Rigid Solids

There are many ways of representing rigid solids. In volume oriented approach of solid model reconstruction, final object is obtained by performing mainly sweep operations and regularized boolean operations which are typical of CSG schemes. So here these two representations are discussed and a brief introduction to engineering drawing is given.

2.2.1 Constructive Solid Geometry (CSG)

Constructive solid geometry [27] connects a family of schemes for representing rigid solids as Boolean Constructions or combinations of solid components via the regularized set operators. CSG representations are best-understood and currently most important representation schemes for solids.

CSG Trees

CSG Tree as shown in the figure 2.3 is a binary tree. Non-terminal nodes represent operators, which may be rigid motions or regularized union, intersection or difference. Terminal nodes are either primitive nodes or transformation leaves, which contain the defining arguments of rigid motions.

Properties of CSG Schemes

CSG schemes are unambiguous but not unique. The domain of a CSG schema depends on the generalized operators used.

If the primitive solids are bounded, then any CSG scheme is unambiguous but not unique. The domain of a CSG schema depends on the available operators.

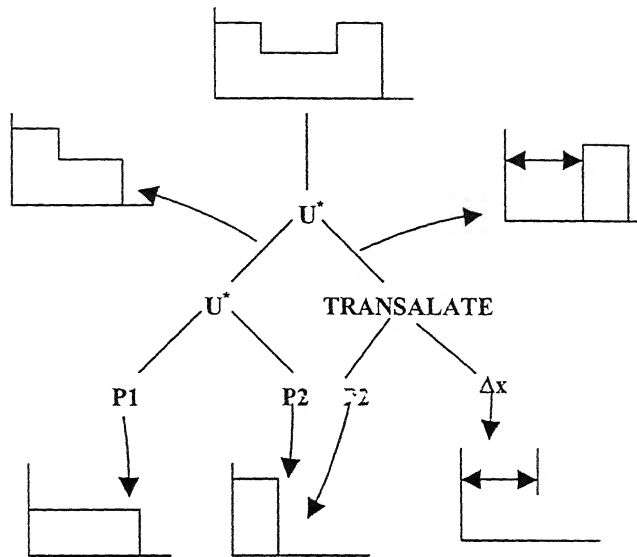


Fig. 2.3 A CSG Tree and solids represented by its sub tree

When the primitive solids are bounded, then a CSG tree is valid if the primitive leaves are valid. Because the primitive leaves validity is easy to find out, validity of a CSG tree can be very easily decided.

Thus, the scheme depicted in figure 2.4 has the same domain if both schemes have general motion and combinational operators.

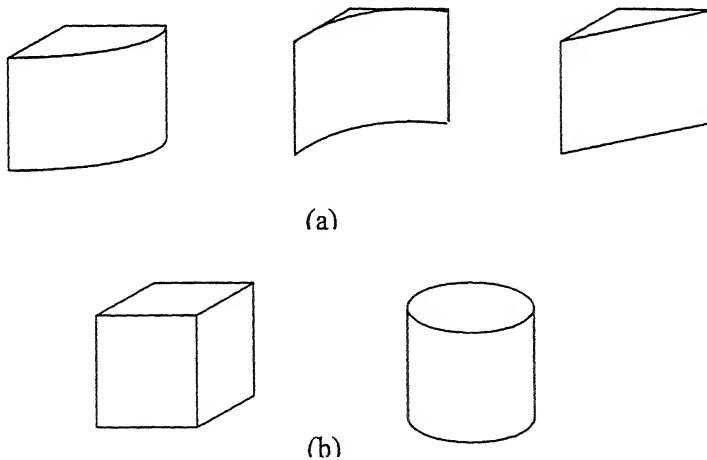


Fig. 2.4 Two CSG schemes having different primitives but same domain

CSG schemes whose primitives match the domain of the represented objects concise. Schemes based on the bounded primitives usually are more concise than those based on general half spaces.

CSG representations are not efficient source of geometric data for producing line drawing of objects and certain types of graphic interactions like picking an edge. However it is easy for generating shaded displays and computing integral properties of solids.

2.2.2. Sweep Representation

The basic notion embodied in sweeping scheme is very simple: A set moving through a space may sweep a *volume* that may be represented by the *moving object* plus the *trajectory*. However sweeping is the least understood of all the schemes.

Transalational Sweep

Consider a 2-D set, A lying in a plane and line segment B perpendicular to the plane of A and having end point on the plane. Let S be the solid swept by A and it is translated, parallel to its plane, along the trajectory B. So, representing A and B may represent S. So, representing the 3-D solid S is reduced to that of a 2-D plane, A.

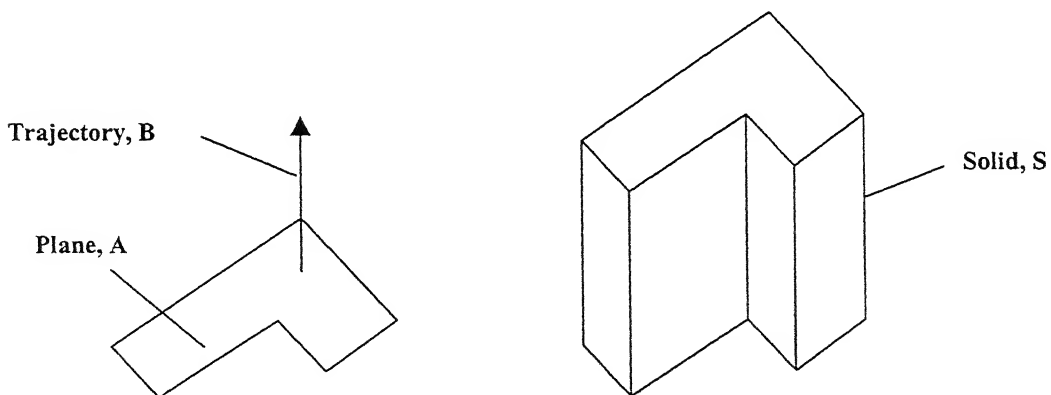


Fig. 2.5 Transalational Sweep

Rotational Sweep

This is analogous to translation sweep; a plane, A called Generatrix revolves around an axis, B sweeping a solid. Axis is generally coplanar with the generatrix.

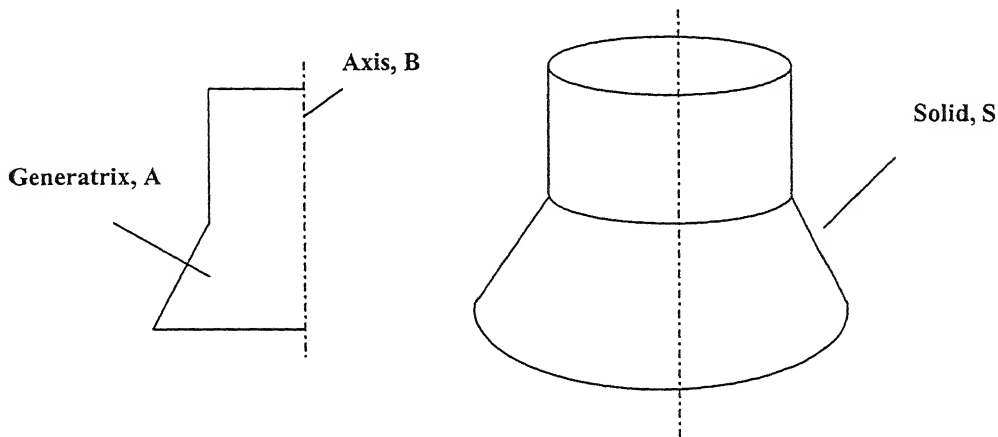


Fig. 2.6 Rotational Sweep

2.2.3 Engineering Drawings

Engineering drawings, otherwise known as orthographic projections, are commonly used to describe a 3-D solid object in two dimensions. It is the accurate description of shape where in an undistorted image of the object appears in a flat, transparent, but imaginary projection plane.

Detailed description of engineering drawings is given in Chapter 3.

Chapter 3

Knowledge of Engineering Drawing

An engineering drawing forms the graphic language from which a trained person can visualize the required object. Since an engineering drawing displays a precise picture of the object to be produced, it conveys the same object to every trained eye. Drawings that are prepared in one country can be utilized in any other country, irrespective of the language spoken there. Hence, this graphic language is called the universal language of engineers.

3.1 Importance of engineering drawings

With the completion of the designer's work, the general structure of the machine is defined. The next phase is concerned with the problems of its manufacture. Each part must be constructed as a separate project and later assembled in to a machine. This demands the preparation of a separate *recipe*, which completely describe each piece of the machine and directs the process of its construction. It has long been recognized that a drawing is the clearest and simplest medium through which all mechanical parts can be easily described. Figures, notes and specifications will supplement the drawings. Thus, a drawing gives the detailed information need to manufacture any machine part and a workman who manufactures that part needs no other information regarding it. A typical engineering drawing is shown in the figure 3.1

3.2 Elements of engineering drawings

As explained above an engineering drawing consists of set of all those things required to manufacture individual components. So, typically it consists of

- Geometry information of the object
- Knowledge about the object through various symbols

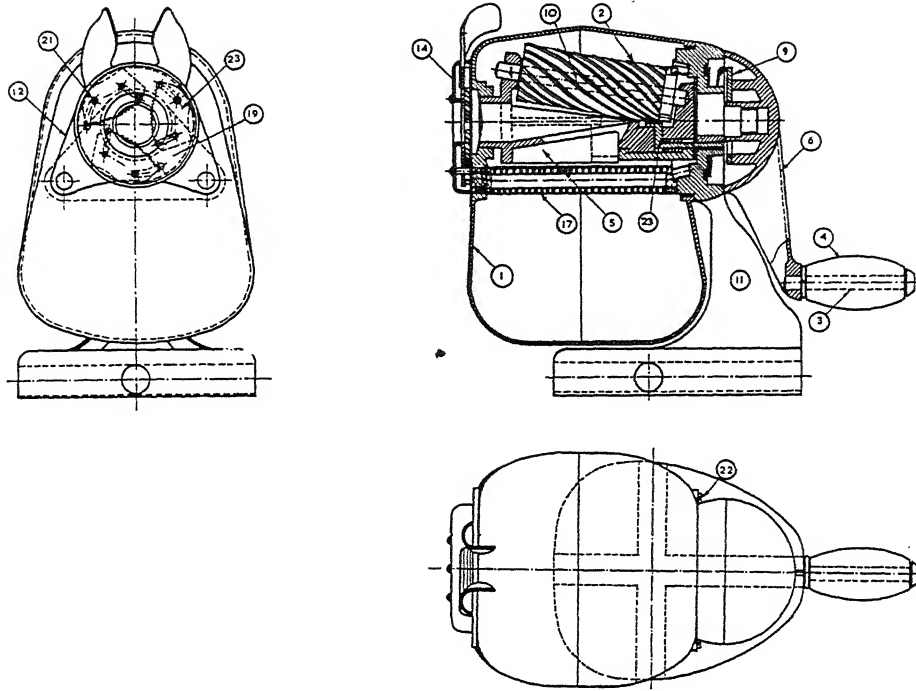


Fig 3.1: A typical engineering drawing

3.2.1 Shape description

Any object has three dimensions, viz., length, breadth and height. The problem is to represent all these three dimensions together with the other details of the object on a sheet of paper with only two dimensions. One method followed is to imagine the object to be projected on to a number of planes, the number being dependant on the complexity of the shape of the object.

A *projection* is defined as a view imagined to be projected on to a plane known as the projection plane. Projection plane is only an imaginary plane. From every point on the object, parallel lines called *projectors* are drawn perpendicular to the projection. According to the position of the plane of projection these view are divided in to two types.

- Orthographic projections
- Auxiliary projections

a) Orthographic Projections

When the projectors are all perpendicular to the plane of projection, the drawing obtained is called *orthographic projection*. The plane of projection is horizontal, vertical or an auxiliary vertical plane. This method has been long established worldwide as the standard practice for shape description on working drawings. To obtain the true shape and proportion of the object

1. The projectors should be parallel to one another and perpendicular to the plane of projection
2. The object must be viewed from such a position that the projectors meet the surfaces to be described, at right angles.

In order to obtain the projections it is usual to imagine the object is enclosed in a rectangular box made of transparent plane sheet of glass box with the principal face of the object parallel to the frontal plane of the projection as shown in figure 3.2

All the sides of the box except the rear face are assumed to be hinged, to the frontal plane and are opened out so that all the views lie in the same plane. Thus, we will get the multi planar orthographic views on one flat sheet as shown in the figure 3.3. Generally rare and one of the side views are redundant and so are omitted.

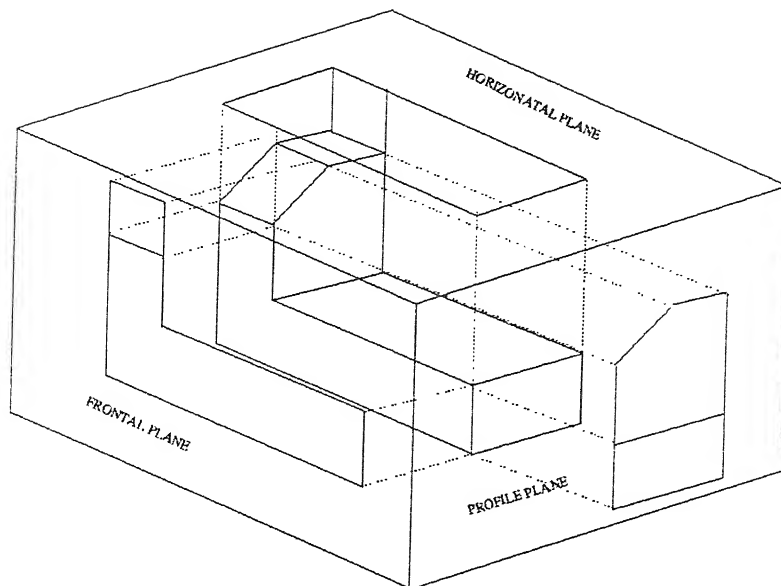


Fig 3.2: The glass box

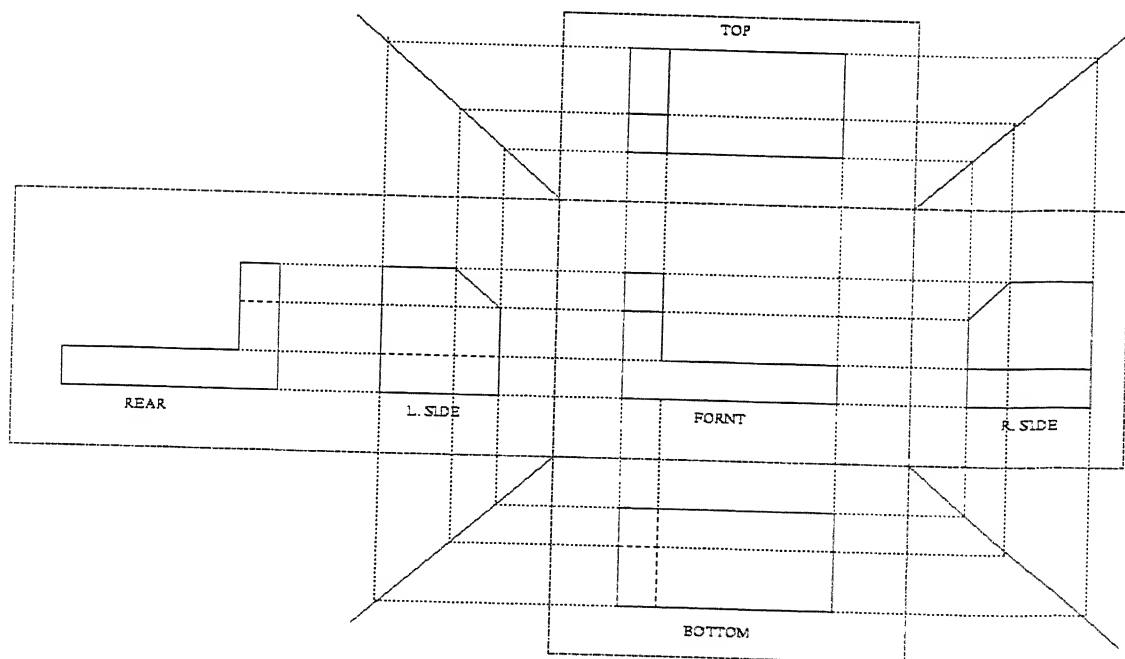


Fig. 3.3: The glass box opened out

More than one orthographic view is necessary

In a detailed drawing, a complete as well as accurate shape description is necessary. In order to show true shape in an orthographic view, the projectors must always be perpendicular to the principal surfaces and therefore any one view is restricted to depict two systems of dimensions only. Hence more than one projection is necessary to complete the description of the object.

The front view of the object gives its length and height, top view gives the length and breadth and side view gives breadth and height.

Types of orthographic projections

There are three principal planes of projections usually at right angles to each other called frontal plane, horizontal plane and profile plane. Both frontal and profile planes are vertical. They are assumed to extend indefinitely and intersect at right angles to each other forming three straight lines called coordinate axes. Frontal and horizontal planes meet at x-axis. Frontal and profile planes meet at y-axis. Horizontal and profile planes meet at z-axis.

Origin is the point of intersection of these coordinate axes. Four quadrants are formed by the intersection of frontal and horizontal plane. These are known as first, second, third and fourth quadrants. The position of the object in these planes give two different types of projections namely the first and third angle projections.

i. First angle projections

The projection obtained when the object is between the observer and plane of projection, i.e., the object is in first quadrant is called *first angle projection*. This method results in showing the right side view on the left side of the front view and top view at the bottom of the front view.

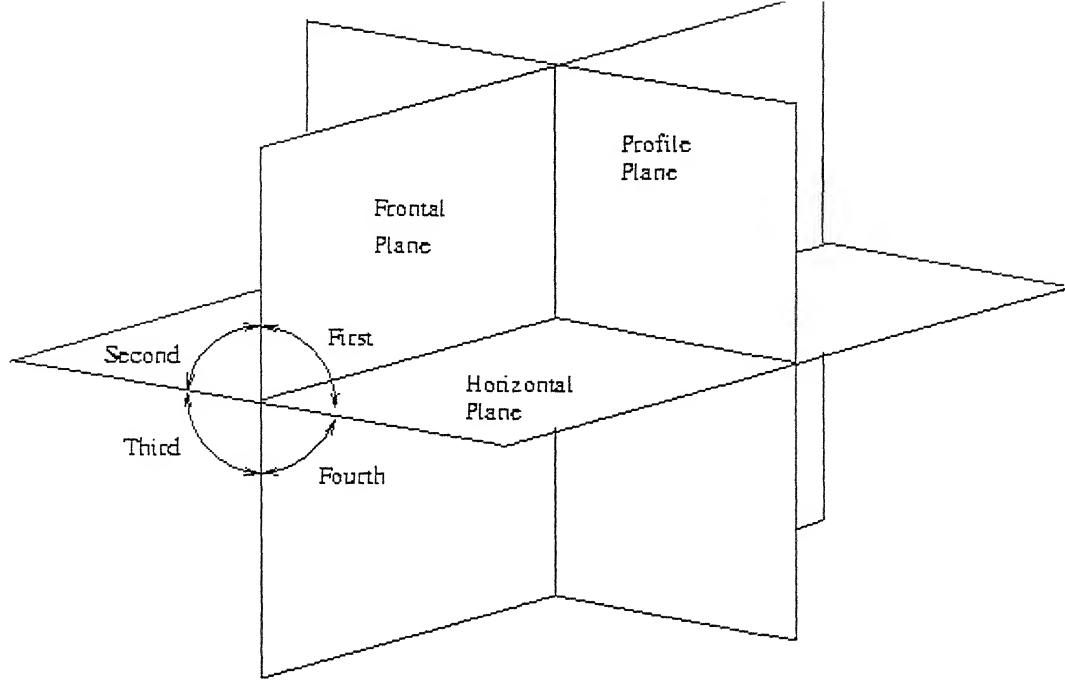


Fig 3.4: The principal planes of projection

ii. Third angle projection

The projection obtained when the plane of projection is between the observer and the object, i.e., the object is in the third quadrant is called *third angle projection*. This method results in showing the right side view on the right of the front view, top view on the top of the front view. Thus, the views are placed in their natural positions.

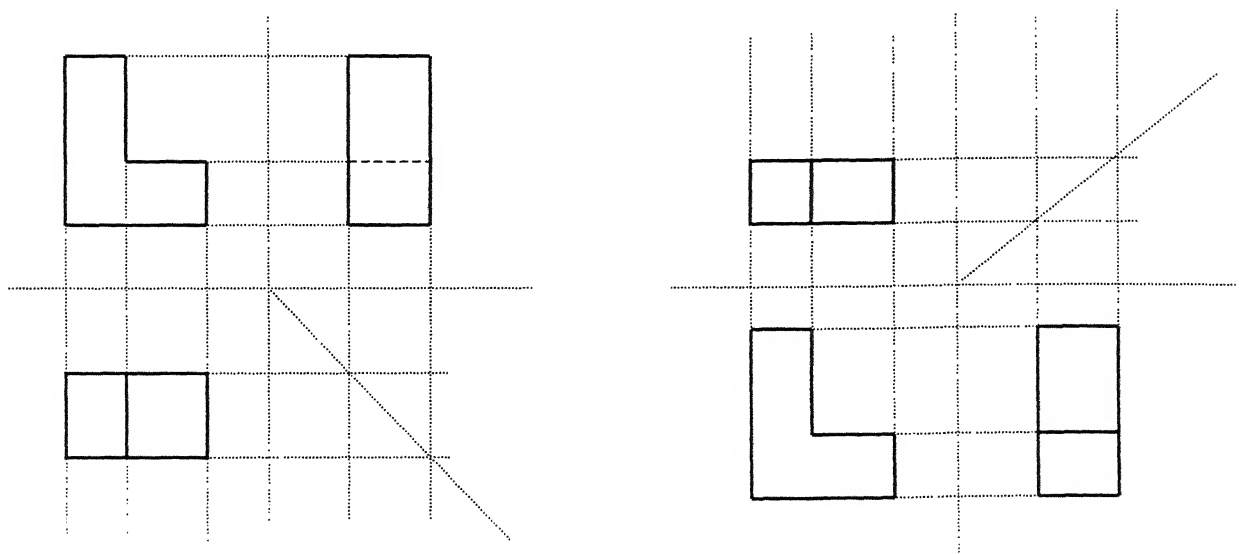


Fig. 3.5: First Angle and Third Angle Projections

Any of these methods can be used to describe the object completely. First angle projections are widely used in India and Europe, where as third angle projections are widely used in USA.

b) Auxiliary projections

Some times the conventional front, top or side views may not be sufficient for shape description of an object with irregular surfaces that are inclined to two or more of the coordinate planes of projection. The true shape of such an inclined surface can be obtained by projecting it on to a plane, which is parallel to it. This imaginary plane is called *auxiliary plane* and the view obtained is called *auxiliary view*.

An auxiliary view is usually partial view showing only the inclined surface. This is sufficient because on auxiliary view showing the entire object is normally confusing in nature and adds nothing to shape description.

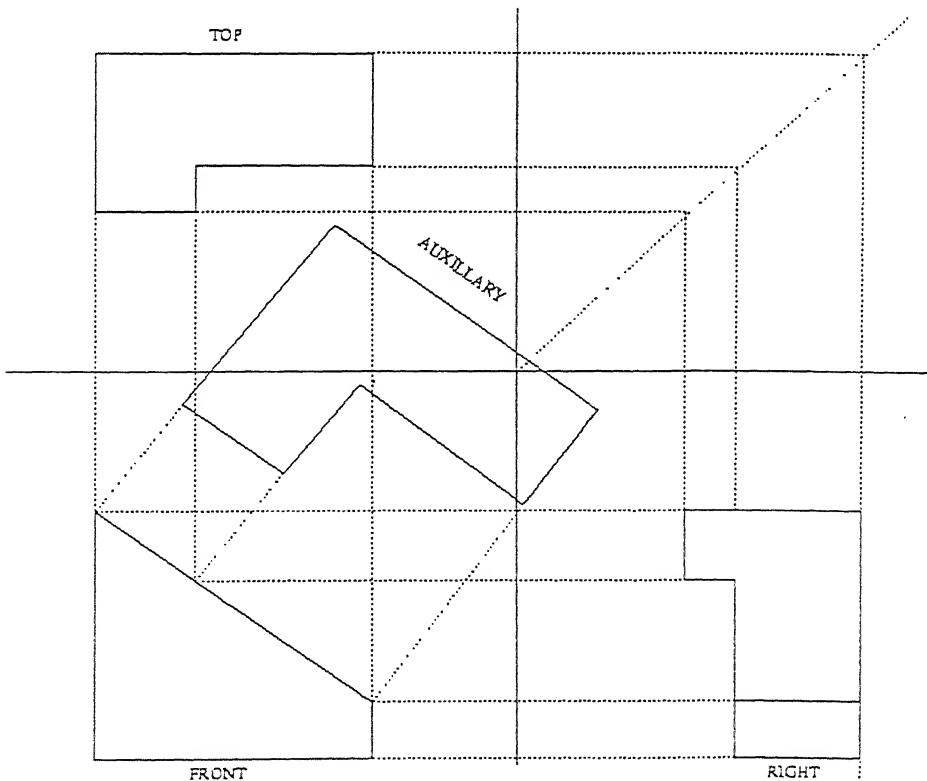


Fig 3.6:Auxillary view of a wedge block

3.2.2 Understanding the Geometry information

In some ways geometry is a type of knowledge base. Geometry of the object is represented by its views. Although, the drawing is completely made up of lines there are certain rules by which one can interpret it. The knowledge base about the geometry is a set of rules that help a trained person like an engineer to correctly interpret about the three dimensions of the object.

An orthographic view contains different types of lines by which an engineer can easily understand its true shape. If a dotted line information is given it means it is conveying the inside information of the object. Humans, if trained can interpret these symbols correctly.

3.2.3 Understanding other information

With the information given by the geometry one can identify the true shape of the object. But there is some other information, which is as important, intact some times more important than the geometry information. Understanding a drawing means understanding the manufacturing process involved, type of surface finish, and type of the materials used. With out this information a drawing will be of no use. To manufacture a part an engineer certainly needs no other information except this data. This knowledge is conveyed in the form of

- Symbols
- Annotations
- Dimensions
- Data base

Symbols

Various symbols tell an engineer about the tolerances followed while manufacturing the object, type of the materials used. Typical mechanical parts such as thread, gears, etc are represented by some standard symbols. Drawing one of its tooth and its nominal diameter

represents a gear. By drawing a full circle and a $\frac{3}{4}$ of circle inside this will represent internal threading. These symbols used in the drawing are standardized.

Drawing these parts in their true shape is time consuming and not economical and so is avoided. There are standard symbols followed whole over the world. So drawings made in one part of the world can be used with out any difficulty in other parts of the world.

Annotations

Annotations used in a drawing help an engineer to correctly identify the semantics. If hidden line information is given that means it is telling about the inside information of the object. If a chain line is given it represents a section plane. Like this there are some other line symbols each having its own meaning. By sectioning the inside details of the object are clearly understood. Sectioned parts are represented by the method of *hatching*. The type of hatching tells the material used. So by sectioning an object we can know the inner details as well as the material used.

Dimensioning

Size description otherwise known as *dimensioning* must be added to the definition of the shape of a machine part in a drawing. Even though the views may be drawn to scale, it is inefficient to require the workmen who construct the piece to measure the drawing. Further more, such a practice would not assure the necessary precision of manufacture. Original drawings are never used in the shop, but are reproduced for such distribution. In the printing process slight alterations in scale may well occur.

Sizes must be stated in figures upon a drawing for the convenience of the workman as well as in the interest of accuracy. These statements of size are termed *dimensions* and are comprised of figures stating the size and lines that serve to explain the correct reference or extent of the measurement.

Dimensioning is made for the sole purpose of adding the workman in the efficient and accurate construction of the piece. The information he requires should therefore be

presented not only in the clearest manner but also in a form that is directly usable without any interpretation.

All the engineering objects have their own way of dimensioning. Dimensions are generally made by drawing two thin parallel lines from the edges to be dimensioned and a line with two arrowheads is drawn between those parallel lines. Dimensions of radii proceeds with a letter **R**. Dimensions of threads starts with a letter **M**. Dimensions play an important role as they are immense help to the workman involved in manufacturing the object.

Data Base

Drawings will also carry a database with them. Drawing of a complicated machine part always contains a database. Drawing database typically contains the number of machine parts needed to manufacture the object. This database typically contains number of the bolts used, number of subparts incase of complicated assemblies etc.

3.2.4 understanding engineering drawing by computer

Humans can understand the drawings by interpreting the symbols and annotations used in it. All humans can't understand a drawing. A person has to be trained. The knowledge of the various notations used has to be imparted to him before he can interpret a drawing.

It is the case with computer also. Knowledge of the symbols used in a typical drawing has to be imparted to it. More the knowledge base of the computer more will be its ability to interpret a drawing.

Neural networks are used to train a computer to understand a drawing. Some neurons are trained to interpret the data of an engineering drawing. Neurons are trained dynamically also. As and when some new information is obtained, these neurons are trained to add that knowledge in to the database.

A drawing is typically in the form a picture. Computer understands it only by the difference in the intensity of the pixel information in the image. To understand the

geometry involved, first it has to be converted in to vector form. A vector data typically contains the length between two points and also its direction. Converting the image form of data (raster data), in to vector form is done by some of the standard image processing algorithms. Once, the data in raster for it can be processed to understand the drawing.

By finding the connectivity by lines and arcs in the drawing the geometry information can be found out. By suitable training of the neurons other symbols like hatching, threads and tolerances etc., can be identified. Every engineering drawing also contain some hand written letter and symbols, that information is to pass information to others who sees those drawings. This information has also to be identified.

In present work, an attempt is made to interpret the geometry of the drawing by the knowledge of representing 3D objects in two dimensions. Hidden line information is also processed.

Chapter 4

The Algorithms

The main underlying principle of this algorithm used is to view the general polyarc formed by an orthographic view as being composed of elementary objects such as arcs, triangles and quadrilaterals. These elementary shapes have to be obtained from the orthographic view. Once these elementary shapes are obtained, a 3D object can be formed using the knowledge of their 2D representation.

4.1 Knowledge of 2D representation for 3D objects

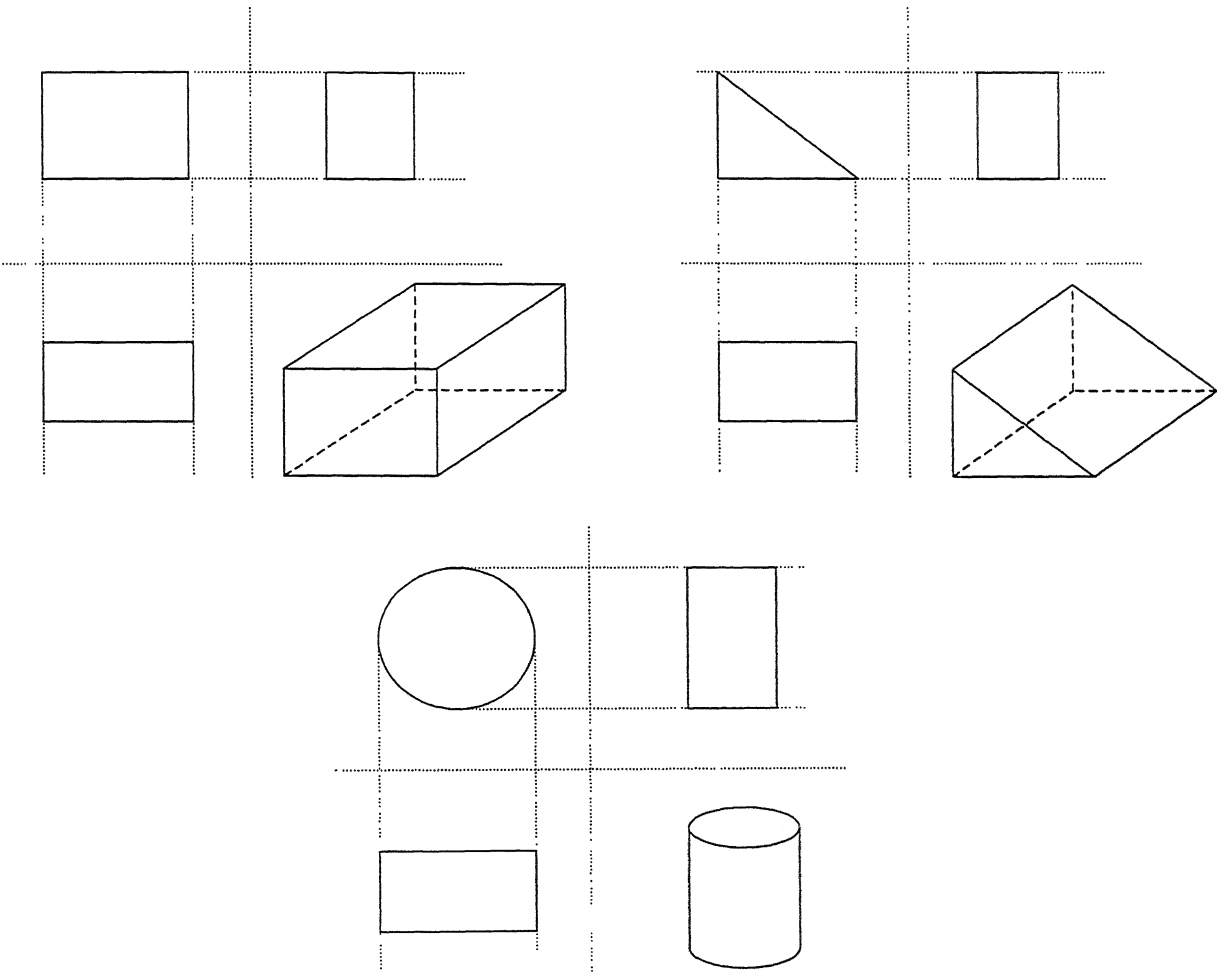


Fig. 4.1: Knowledge of 2D representation for rectangular prism, triangular prism, and cylinder

If we take a rectangular prism, its front, top and side views will be of the form of a rectangle. Front view gives its width and height, top view gives its width and depth and side view gives its height and depth. If a rectangle is formed in front view with its view also being a rectangle having the same height as that of the rectangle in the front view and other rectangle is found in the top view whose width being same as that of the rectangle in front view and whose height being same as that of the width of the rectangle in side view, then it can be said that all these views represent a rectangular prism.

Similarly if we find a triangle in one view and two more rectangles in other views and if their dimensions match, as that of the previous case we can say that it corresponds to a triangle prism. In the same way if we find a circle in one view and two rectangles in other two views and their dimensions match as that of the previous case, we can say that it is cylinder.

But seldom we will find these shapes directly in engineering drawings. So, in this section the process of obtaining these basic shapes from the generalized polyarc, which represents an orthographic view, is explained.

4.2 The input

In this thesis input is fed manually by reading the orthographic views. Point in the view will be referred as node hereafter. Reading the view is basically reporting the connectivities with the entity being a line or an arc with the coordinates specified for each node of the connectivity. A typical file specifying one of the orthographic views, which is shown in figure 4.2, is explained in appendix.

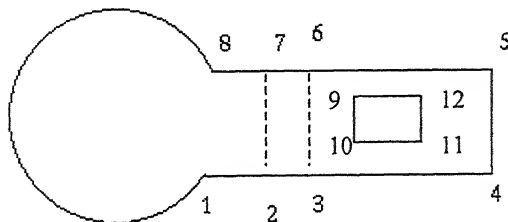


Fig. 4.2: A typical orthographic view

4.3 The loop

A loop is a continuous traversal from a node to same node through a set of nodes. By virtue of its definition it is a closed entity. The view represented in figure 4.2 has 7 loops. The intension to extract a loop is to have regular polyarcs from general polyarc, which is represented as an orthographic view.

The loops which when extracted are shown in fig. 4.3 (a) to 4.2 (g)

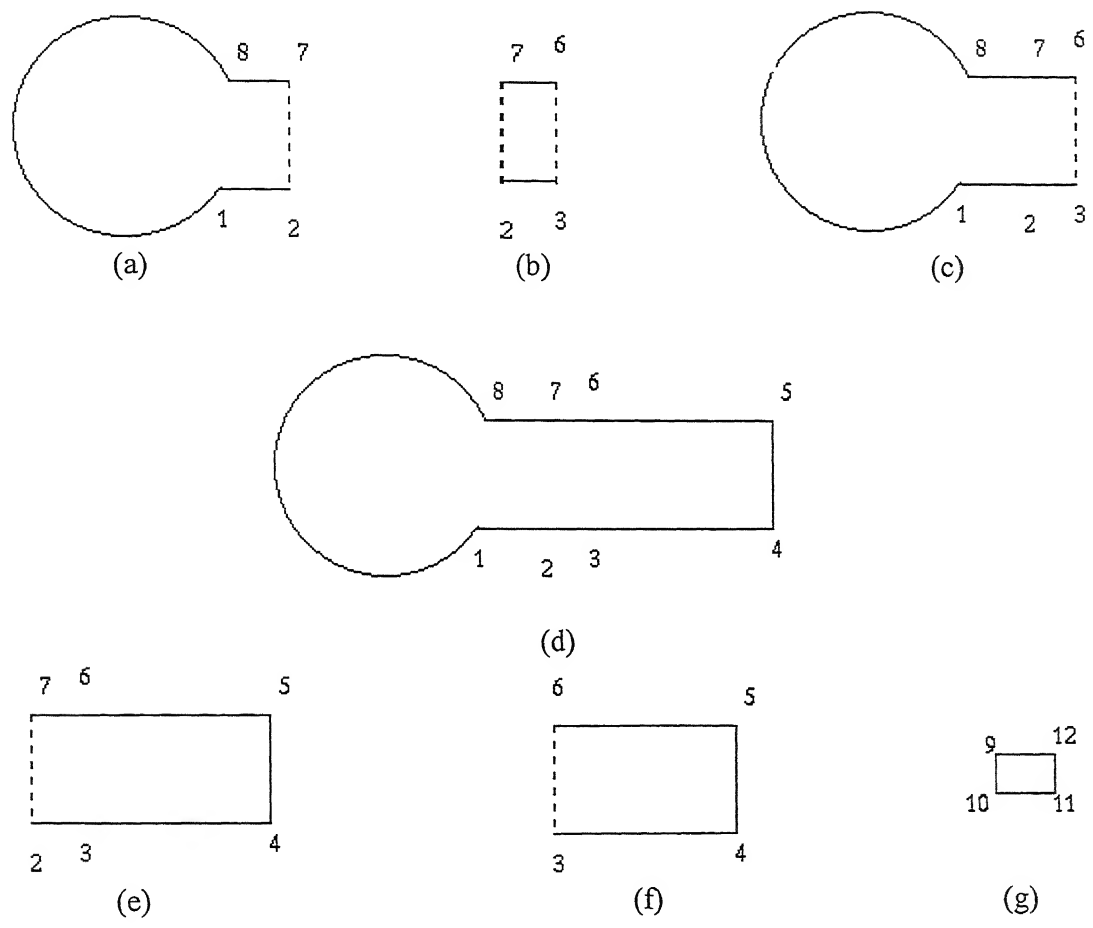


Fig. 4.3: Loops formed from the orthographic view

4.4 Extraction of the loops from orthographic projections

Input is supplied in the form of drawings. A drawing is a general polyarc. The objective is to interpret these polyarcs, which are the orthographic views of the solid in consideration. The algorithms presented in this section describe the way to convert these orthographic views, which are general polyarcs, to a set of regularized polyarcs.

4.4.1 Tree structure

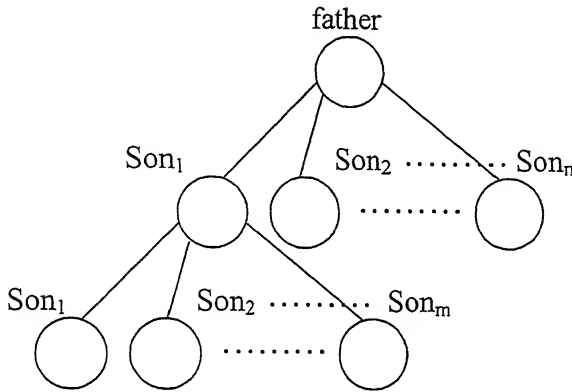


Fig. 4.4: A Tree structure

The tree structure shown in figure 4.4 is used to store loops in the current algorithm. Tree starts with a single node and it can have any number of sons say n . A *son* is a node, which succeeds a predecessor, and that predecessor is called *father*. Each son of the tree again can have any number of sons say m . Each node of the tree can only have one father. All sons having the same father form a generation. A tree can spread to have any number of generations at every level, depending upon the problem under consideration.

4.4.2 Building Tree

Starting from any of the node in the view as the starting point of a tree, branches have to be developed. If the first node has n neighboring nodes, $n-1$ branches have to be made.

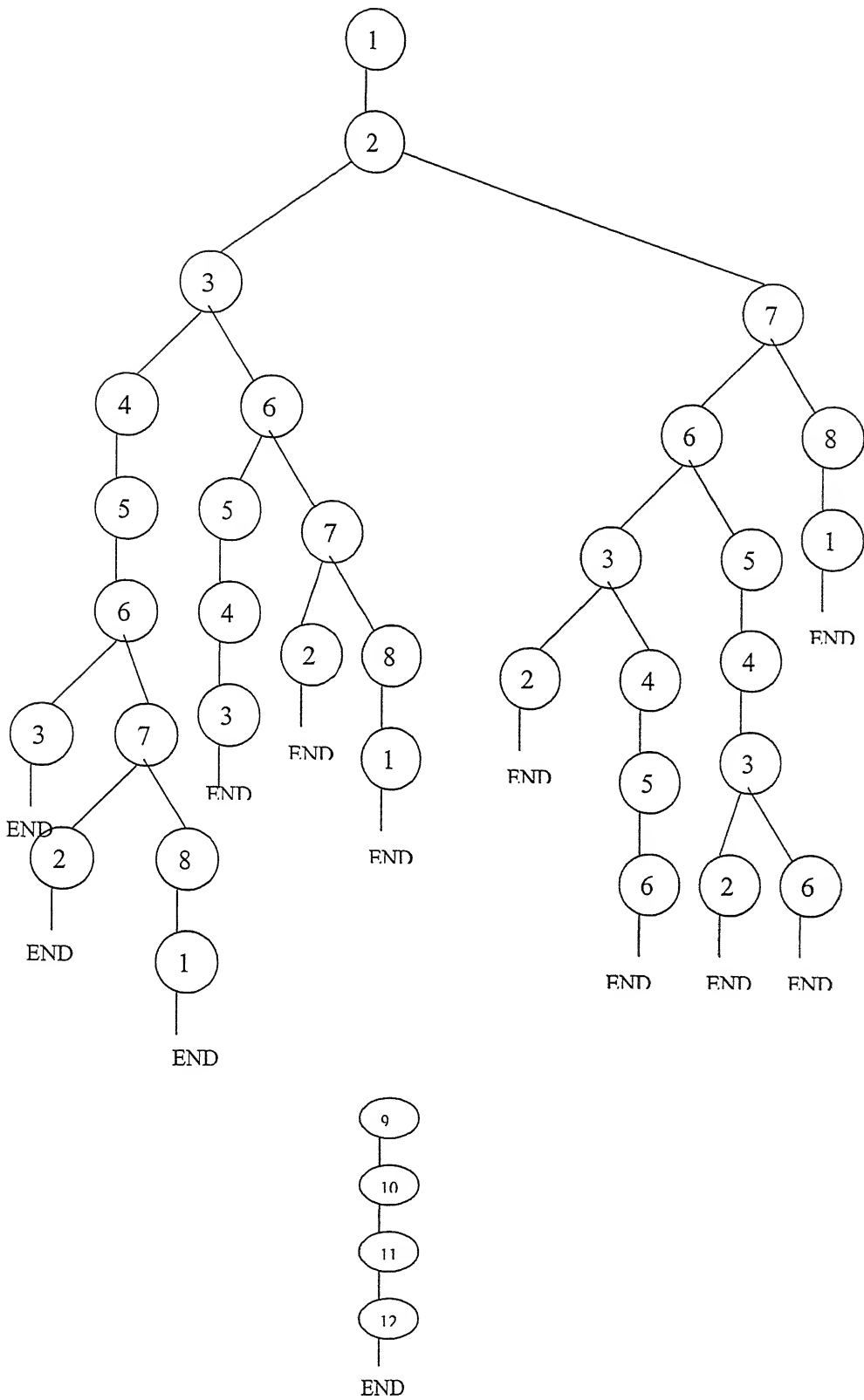


Fig. 4.5: Tree structures formed from the orthographic view in figure 4.2

We can omit one neighboring node, as it will result only in giving repetitive loops. For each son of the starting node, m is the number of its neighboring nodes, then $m-1$ sons have to be created. Here we will omit that neighboring node, which is its father. Then for each son again if n is the number of its neighboring nodes, $n-1$ sons will be created omitting its father node. This process is continued recursively until a closed loop is encountered. Whenever a son is created, the tree is traversed upside towards the first node. On its way, if a node gets repeated then it means that a closed loop has been formed. The tree branches ends and the present node will not have any subsequent generations. This process is repeated until every branch of the tree ends.

This process will result in forming all the loops than can be formed by the nodes, which are connected to the starting point of the tree. If some node of the view is not connected to the starting node, loops that can be formed with that node will not be made. So, again another tree structure has to be formed starting with one such node. This process is continued till all the nodes in the view (polyarc) are considered. The output of this step is shown in fig 4.5.

4.4.3 Obtaining the loops from the tree

Starting from all those nodes, where there is no subsequent generation, the tree is traversed in upward direction until one comes across the same node number as the starting node. These node numbers form a loop. Loops can be formed from all the branches of the tree.

This process yields some repetitive loops. The tree structure shown in the figure 4.5 resulted in 12 loops but there are only 7 distinct loops as shown in figure 4.3. These extra loops have to be removed. The resultant loops from this process may come in clockwise (CW) or anticlockwise (ACW) direction.

The nodes 2,3,7,6 in the case shown in the figure 4.3(b) form a rectangle. But the tree structure yields loops 3-6-7-2, 7-6-3-2. Only one of those two has to be considered. But both of them appear to be different. So as a starting step of a new strategy, all the loops have to be made having the same sense. Here, they are made anticlockwise.

4.4.4 Making a loop anticlockwise

The algorithm implemented for obtaining the loops from the free as explained in the section 4.4.3 would either yield a clockwise loop or anti clockwise loop. Clockwise loops can be converted to a anticlockwise one using following steps.

A loop shown in the fig. 4.6 is a clockwise loop. Tree structure gives it as 1-7-2-6-5-3-2. First its direction has to be found. The direction of the loop depends on the sum of the determinants

$$\begin{vmatrix} x_1 & x_7 \\ y_1 & y_7 \end{vmatrix} + \begin{vmatrix} x_6 & x_5 \\ y_6 & y_5 \end{vmatrix} + \begin{vmatrix} x_4 & x_3 \\ y_4 & y_3 \end{vmatrix} + \begin{vmatrix} x_2 & x_1 \\ y_2 & y_1 \end{vmatrix}$$

Where, x_1, x_2, \dots, x_7 and y_1, y_2, \dots, y_7 are x, y coordinates of points 1,2,3...7

If the value of this determinant comes out to be positive, then the loop is anticlockwise; else it is clockwise. To make it anticlockwise keep the first node of loop as it is and then reverse all the remaining node numbers. So, 1-7-6-5-4-3-2 becomes 1-2-3-4-5-6-7.

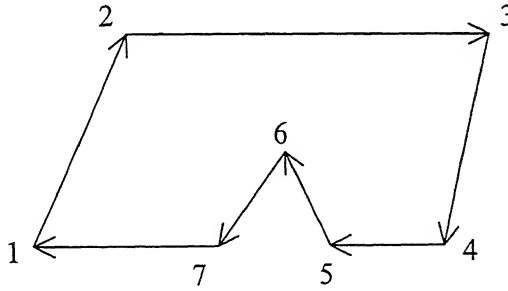


Fig.4.6: A clock-wise connected loop

4.4.5 Deleting the repetitive loops

Consider the case shown in the fig. 4.3(b). After making all the loops obtained from the tree structure anticlockwise; we have 3-6-7-2, 7-2-3-6 corresponding to figure 4.3(b).

Now change the order in such a way that the first node number of the loop is the smallest among them. So, both become 2-3-6-7, 2-3-6-7 and one will be deleted.

4.4.6 Redundant Nodes

Consider the case shown in fig. 4.3(d). In this case node numbers 2,3,6,7 are redundant. At this node polyarc has C^1 continuity. If the slope have at ending point of the first curve segment is equal to slope at the starting point of the second curve segment then these two curve segment are C^1 continuous at this point provided they meet at that point (C^0 continuity). So, these nodes have to be deleted from the polyarc before sending the loop for further processing.

4.5 The Discretization

Every view can be discretized into basic 2-D primitives such as triangles, circular and quadrilateral primitives. Many special cases will arise while discretizing these views. The data of every view has been conveniently stored in the form of loops. Every primitive has its own sign associated with it: positive or negative. A positive primitive means it has to be added finally similarly a negative has to be subtracted. The algorithm presented to discretize is recursive in nature i.e., until it finds a primitive not been extracted from the loop it searches until it is found. All such primitives then represent the data of the views for the next step.

4.5.1 The termination criteria

As discussed above, the algorithm to discretize is recursive in nature. This algorithm basically consists of 4 steps. Each step will also have some special cases. The algorithm can start from any of these four steps. Each step has its own objective associated with it. If even one of the objectives is not met the termination criteria will not be reached. If at any stage of this recursive process the loop under consideration is having less than or

equal 4 nodes then process will be terminated, as a primitive having less than or equal 4 nodes need not be discretized further.

4.5.2 Separation of arcs

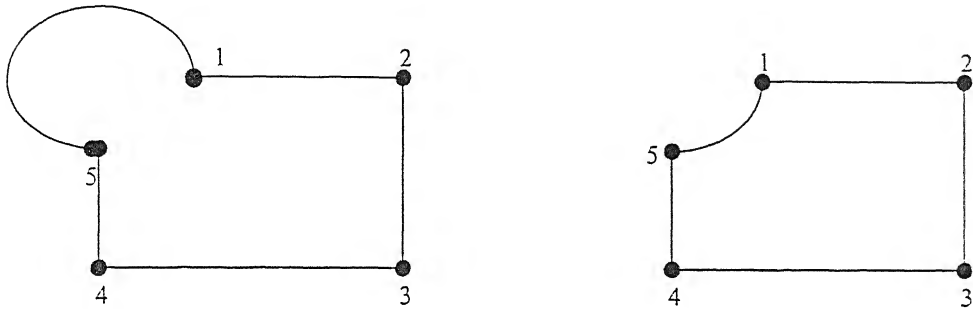


Fig.4.7: Loops with CW and ACW arcs

A loop as shown in figure 4.7 contain arcs. Objective of this step is to separate these arcs from the loops if possible. An arc connects the points 1,5 as shown in the figure 4.7. Join these two points by a line segment. Now polyarc 1-2-3-4-5 becomes polygon 1-2-3-4-5. And arc corresponding to points 1,5 will be another primitive. Now the sign attached to this circular primitive has to be decided.

As the loop is connected in anticlockwise direction, if the arc is also in anticlockwise direction it is positive else it is negative. So, after this step 1-2-3-4-5 polyarc is discretized into 1-2-3-4-5 polygon and 1-5 arc as shown in the figure 4.8

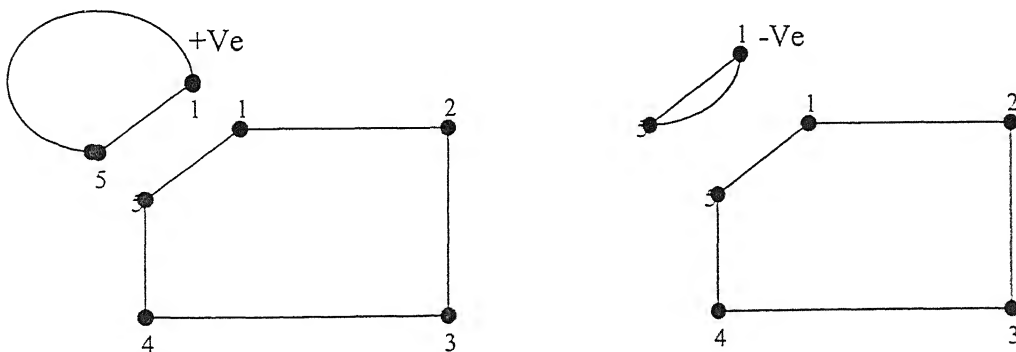


Fig.4.8: Loops shown in figure 4.7 after arc separation

This step should not be performed if the line joining the nodes connected by an arc (nodes 1 and 5 in the case considered) intersects with any other segment (line or arc) of the polyarc or any other nodes of the polyarc lies on this line segment. Every edge of the polyarc is tested for intersections with the line segment 1-5, if any segment is found intersecting with line 1-5, this step of separating the arc is not performed. The same step is performed for other arcs (if any) in the same polyarc. Redundant node found if any has to be deleted. If no more arcs in the polyarc are left, next step of the algorithm can be started.

There is a special case of an arc being tangential to the edges it joins. This special case is explained later in this chapter (section 4.7).

4.5.3 Separation of triangular primitives

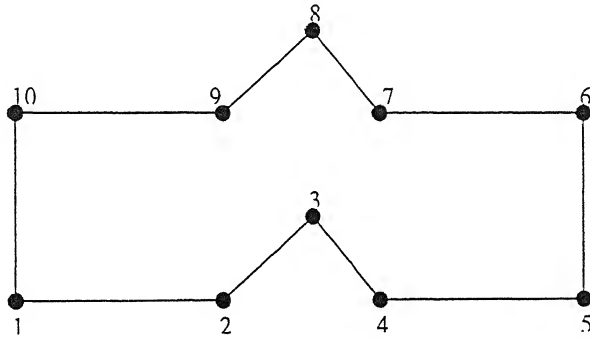


Fig.4.9: Loop having separable triangular primitive

The objective of this step is to separate the triangular primitives, which can be formed directly from the polyarc. Any two line segments form a triangular primitive if both are inclined and have one meeting point. The line segments 7-8 and 8-9 form triangular primitives as shown in figure 4.9. The important step in this algorithm is to join node with node 9 directly. As the loop is anticlockwise, if the signed area of the triangle 7-8-9 is positive, 7-8-9 is a triangular primitive; else if the signed area is negative, then 7-8-9 is a negative triangular primitive. The loop after this step is shown in figure 4.10.

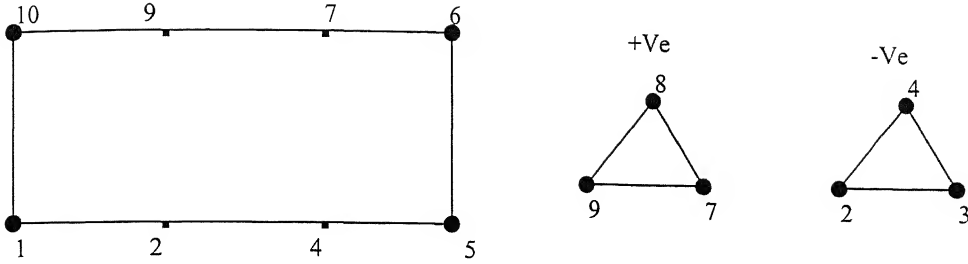


Fig.4.10: Loop in figure 4.9 after triangular separation

If the line segment 7-9 or 2-4 (shown in the figure 4.9) intersects any other line in the polyarc or if any other vertex of the polyarc lies on the line segment 7-9 or 2-4, then this step should not be performed. Once the whole polyarc is processed, redundant nodes (if any) have to be removed. Now this loop data will be sent to the next step in the algorithm.

4.5.4 Separation by ray casting

This algorithm mainly filters out 4-noded elements (quadrilaterals). In cases of complex loops, instead of separating 4-noded elements this step splits the polyarc into two separate loops. Each separated loop will be processed again using the same set of algorithms.

Every thing starts with casting a ray in this algorithm. If ever there is at least a horizontal edge in the original loop this algorithm is implemented. A ray is cast from abscissa $-\infty$ and ordinate same as that of horizontal edge to abscissa $+\infty$ with the same ordinate. The infinity used here ensures that the starting and ending point of the ray does not fall on or inside the original loop (polyarc). The ray is broken in to smaller segments wherever it encounters a node of the polyarc on it. For each line segment of the ray, and whenever it intersects the polyarc line membership classification is done in order to find whether the line segment is on, inside or out of the polyarc. This ray casting while intersecting the polyarc may also introduce a new node at the points of intersection. This will happen if any of the edges of the polyarc are inclined or perpendicular to the ray. An inclination means that a triangular primitive will be subsequently looped out by

continuous implementation of these member algorithms. Similarly an arc will also generate a circular primitive. The only case, which has to be considered, is one where an edge is perpendicular to the ray. In that case the new points (points of intersection) should be added the list of nodes representing the original polyarc.

The polyarc is then traversed from the node, which is the starting node of the line segment of the ray cast, which is inside the polyarc to the ending node of the same line segment of the ray cast in any sense (CW/ACW). This line segment, which is inside, bypasses the polyarc splitting it in to two.

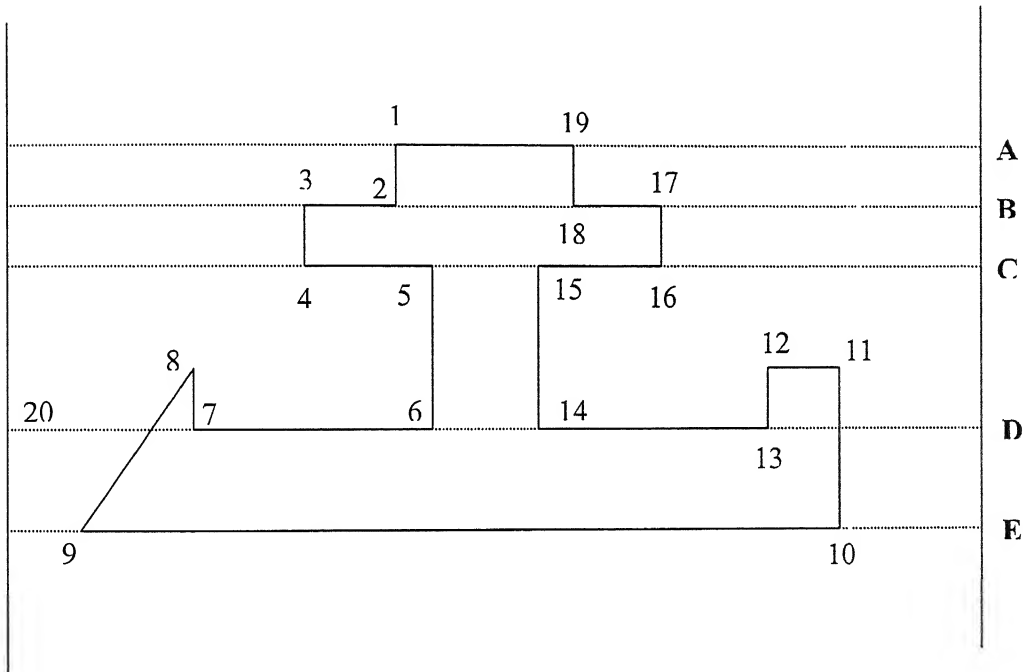


Fig.4.11 Separation by casting a ray

In figure A, B, C, D, E are the rays cast. For the ray and the line segment 2-18 is inside polyarc as shown in figure and the polyarc is traversed from 2 to 18 either through 6 or through 19.

In either way the polyarc is bypassed by the segment 2-18 and thus split in to two. In the same figure the ray B yields a square element and a relatively simpler loop 2 to 18 through 6. But ray C if implemented first will yield two relatively simpler polyarcs. On

the ray D the line segment 20-7 has some part, which is inside the polyarc, it is omitted because of the reasons explained already.

4.5.5 Separating inclined lines

In algorithm defined in section 4.5.3, which precipitates the triangular primitives, a single line segment is sufficient to take out a triangular primitive. In the polyarc, if a single inclined line segment exist it has to be converted in to a triangular primitive by adding two perpendicular line segments to it.

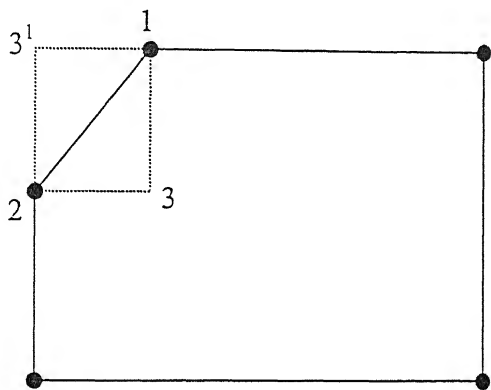


Fig.4.12 Inclined line separation

The length of one of the line segment, which forms the triangular primitive, which is parallel to x-axis, is the absolute value of the difference of the abscissa of the inclined line segment. The length of the other line segment, which is also the part of the triangular primitive, is the absolute value of the difference of the ordinate values of the inclined line segment. Inclined separation may lead giving a positive or negative triangular primitive depending upon the relative position of the point considered for the separation with respect to the inclined line. For making the new triangular primitive either it can be started from point 3 or from 3'. The triangular primitive is traversed along the direction specified by the inclined line segment (1-2) and its signed area is noted. A positive signed area makes the triangular primitive positive and negative area a negative one. The triangular primitive thus chosen should not intersect with any of the other edges of the polyarc so as to make the original polyarc a non-self intersecting one.

Let the starting point of inclined line is 1 and ending point is 2 and the point forming the triangular primitive is 3. To form a valid triangular primitive line segment 1-3 should not have any intersections with other line segments of the polyarc or any other vertex of the polyarc should not lie on this line segment. Figure 4.13 shows the acceptable cases to form a positive triangular primitive.

In case (i) 1-2-3 becomes positive triangular primitive. Now join 1-3 and 3-2 in the original polyarc.

In cases (ii),(iii),(iv) 3 is one of the vertex. Travel from 1 to 3 in the same of the polyarc through 2. Now 1-3 becomes another polyarc, which has to be subsequently sent to the discretization process. In the original polyarc join 1 to 3 directly.

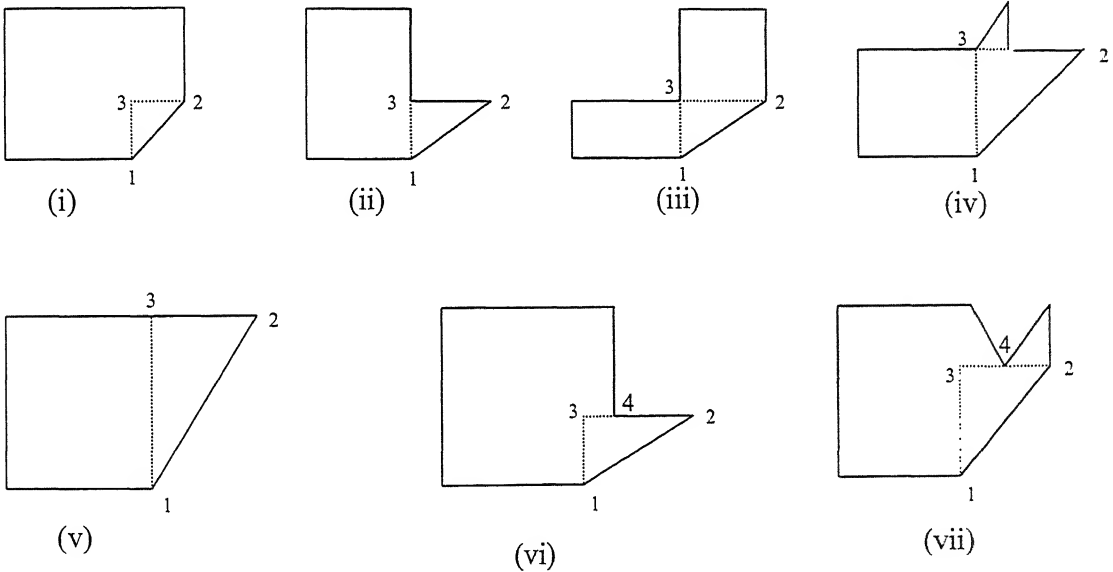


Fig. 4.13: Acceptable cases for forming a positive triangular primitive

In case (v) 3 is not a vertex. So, on the line where 3 is there (2-4) add 3 in between 2 and 4. Travel from 1 to 3 through 2. 1 to 3 now becomes polyarc and sent to subsequent discretization algorithms. Join 1 directly to 3.

In case (vi), (vii) some other vertices of polyarc lie on line segment 2-3. Travel from 1 to 4 through 2. Find a point 4, which is on 2-3 and nearer to 3. Nodes 1 to 4 and 3 form a positive triangular primitive. In the original polyarc join 1 to 3 and 3 to 4.

Let 5 be the point, which is the previous point to 1 on the polyarc. Figure 4.14 shows the acceptable cases when line segment 1-3 will have certain overlap with line segment 1-5.

In case (i) 1-2-3 becomes a positive triangular primitive, join 5 to 3 and 3 to 2 in the original polyarc.

In case (ii),(iii) 3 is a vertex. Travel from 1 to 3 through 2. Primitive 1-2-3 has to be sent for subsequent discretization. Join 5 to 3 directly.

In case (iv) 4 lies on line segment 2-3. Travel from 1 to 4 through 2. 1-4 and 3 becomes a primitive and sent for further discretization. Join 5 to 3 and 3 to 4 in the polyarc.

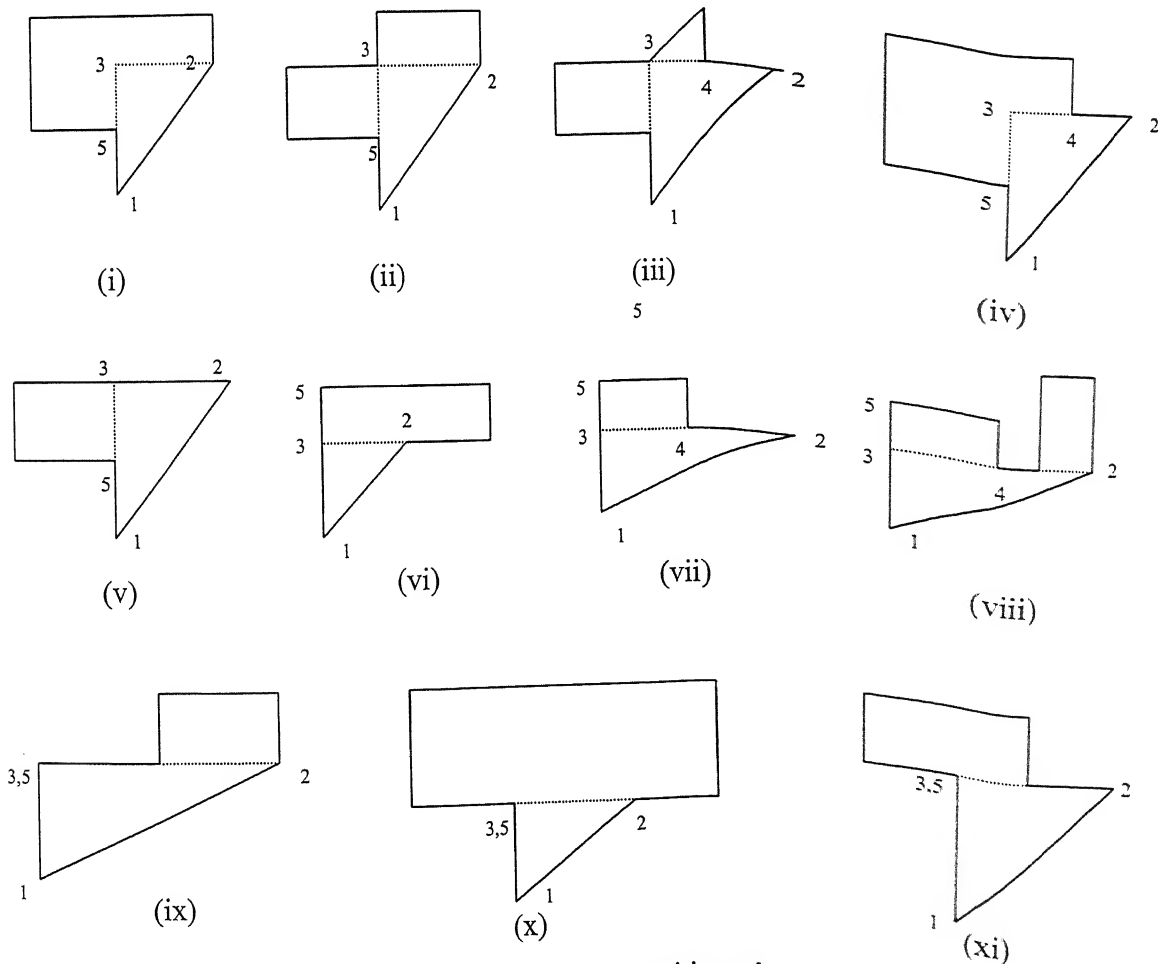


Fig. 4.14: Acceptable cases for forming a positive triangular primitive

In case (v) add 3 on the line segment 2-4. Travel from 1 to 3 through 2 and primitive 1-2-3 is sent for further discretization. Now join 5 with 3.

In case (vi) 1-2-3 is a +ve triangle primitive, join 3 with 2 in original polyarc.

In case (vii) and (viii) 4 is a vertex and it lies on the line segment 2-3. Travel from 1 to 4. 1-4 and 3 is primitive sent for further discretization. Now add 3 in between 5 and 1. Join 3 to 4 in original polyarc.

In cases (ix), (x), (xi) 3 is same as 5.

For case (ix), travel from 2 to 6. 2-6 is another loop; join 2 with 6 in original polyarc.

for case (x), 1-2-3 becomes positive triangle primitive, join 5 with 2.

for case (xi), a vertex 4 lies on line segment 2-3. Travel from 1 to 4 through 2. 1-4 and 3 becomes a primitive and it is discretized further. Join 5 to 4 in original polyarc.

If the case under consideration does not fall in any of the cases mentioned above a positive triangle primitive cannot be formed. So, instead a check for forming a negative triangle primitive is done. Acceptable cases are shown in figure 4.15.

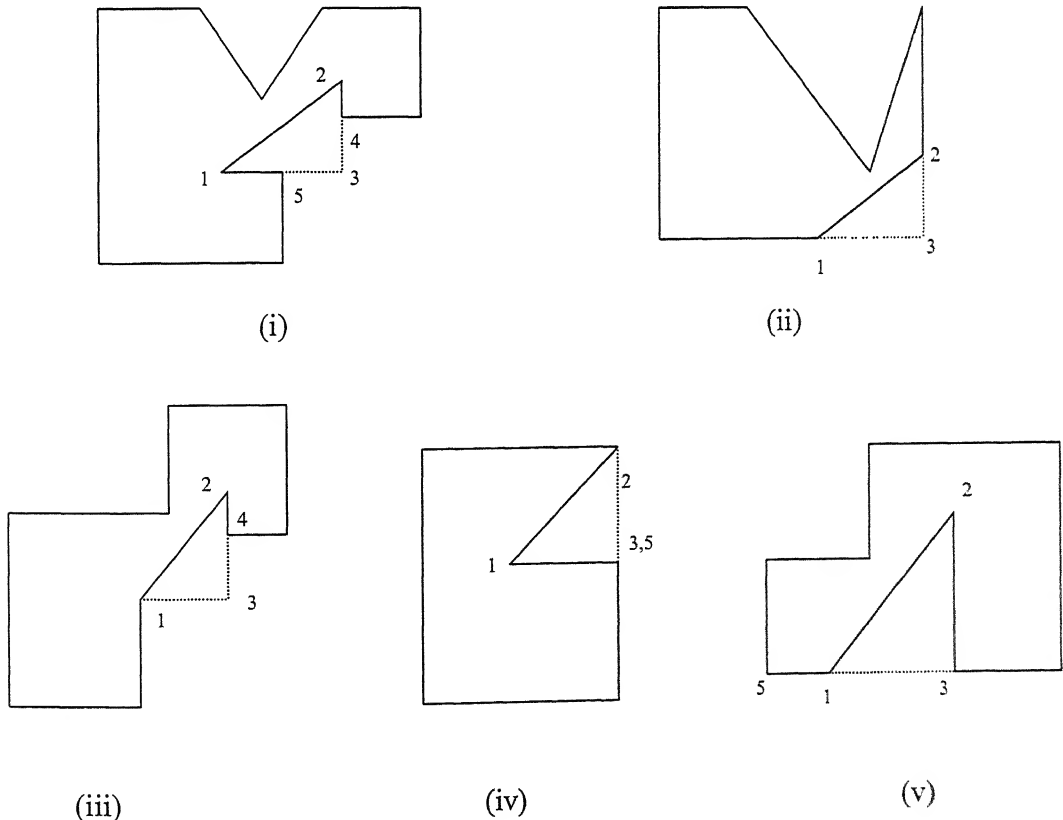


Fig. 4.15: Acceptable cases for forming a negative triangular primitive

No point except 5, which is previous node to 1 in polyarc, should lie on the line segment 1-3 and line 1-3 should not intersect with any other line of the polyarc. Also line 2-3 should not have any intersections with any other lines of the polyarc. Only the point 4, which is next node to 2 in polyarc, can lie on line 2-3.

In case (i) 1-2-3 becomes a negative triangle primitive. Join 1 and 3, 3 and 2 in original polyarc.

In case (ii) 1-2-3 becomes a negative triangle primitive. Join 1 to 3 and 3 to 4 in original polyarc.

In case (iii) when point 5 lies on 1-3. 1-2-3 becomes a negative triangle primitive. Join 5 with 3 and 3 with 4 in original polyarc.

In case (iv) 1-2-3 becomes negative triangle primitive and join 3 to 2 in original polyarc.

In case (v) 1-2-3 becomes negative triangle primitive and join 1 to 3 in original polyarc.

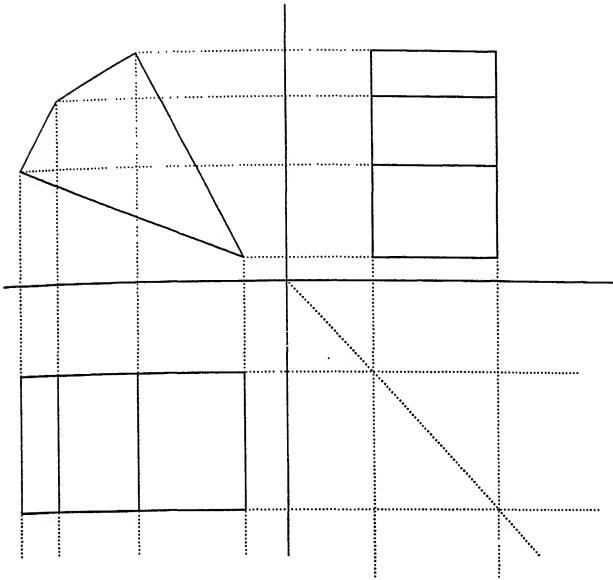
4.6 Constructing 3D primitives from 2D primitives

All the loops formed by the view polyarc have been discretized and are converted in 2D quadrilaterals, triangles and arcs. Objective of this step is to compare these 2D primitives in all the three views and after matching these for correspondence 3D primitives are formed.

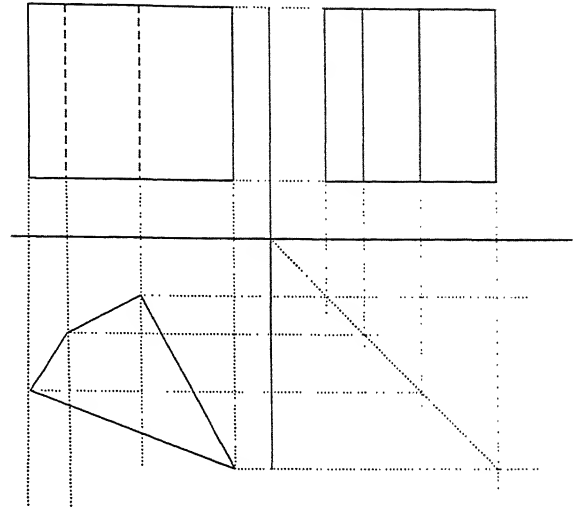
Each of the front, top and side view now contains a set of quadrilaterals, triangles, arcs. These 2D primitives have a sign associated with them either negative or positive. All quadrilaterals have positive sign. Triangles, arcs have either negative or positive.

A quadrilateral in one view forms a set of 3 rectangles in other two views. A triangle in one view forms a set of 2 triangles in other views. If ordinate of any two points in a triangle or quadrilateral are same the number of rectangles are reduced. Depending upon the starting and ending angles of the arc, we get a single rectangle or a set of 3 rectangles in other two views. 2D primitive in one view gives 2 dimensions and its other dimension is obtained by its relative position in the other two views.

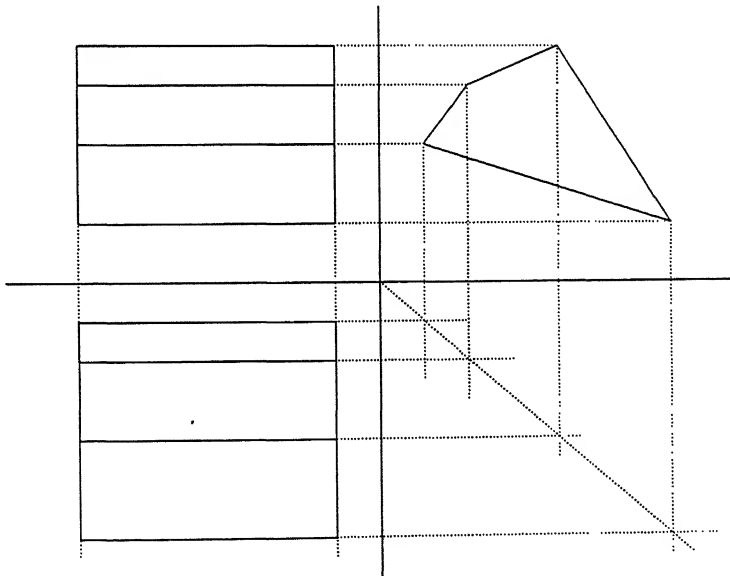
Figures 4.16, 4.17, 4.18 show the cases of quadrilateral, triangle in their front, top and side views.



(a) quadrilateral in front view

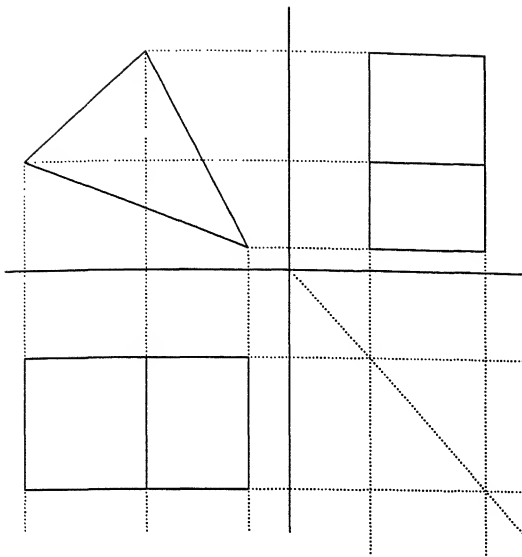


(b) quadrilateral in top view

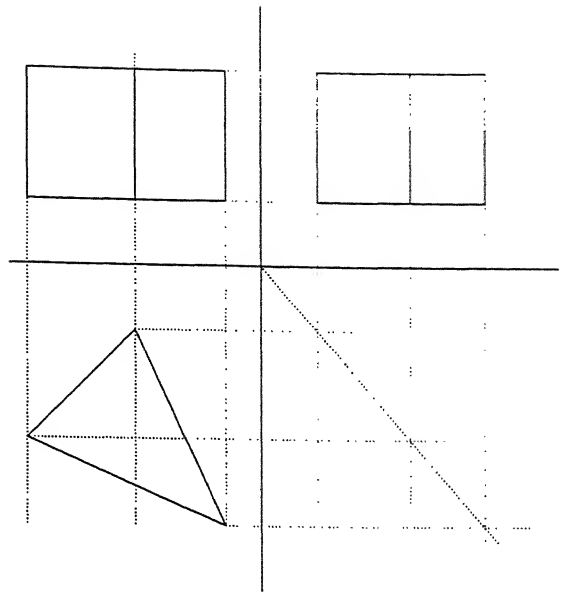


(c) quadrilateral in side view

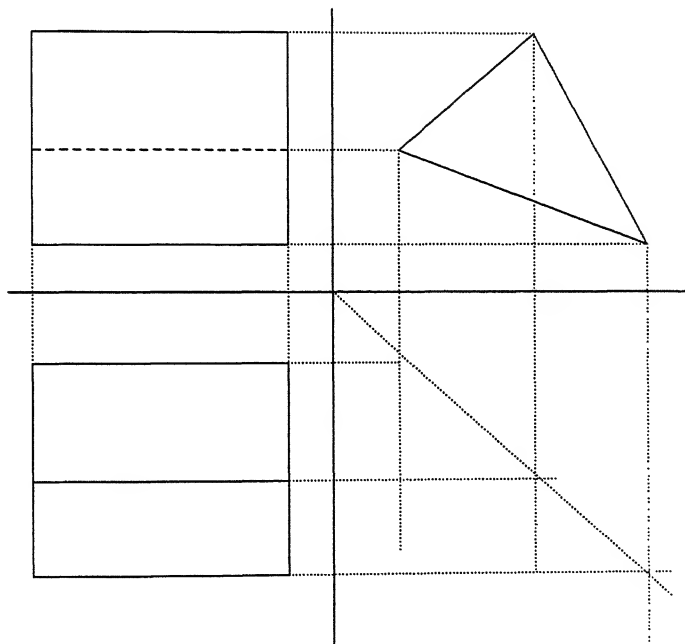
Fig. 4.16 orthographic views of a quadrilateral in various positions



(a) triangle in front view

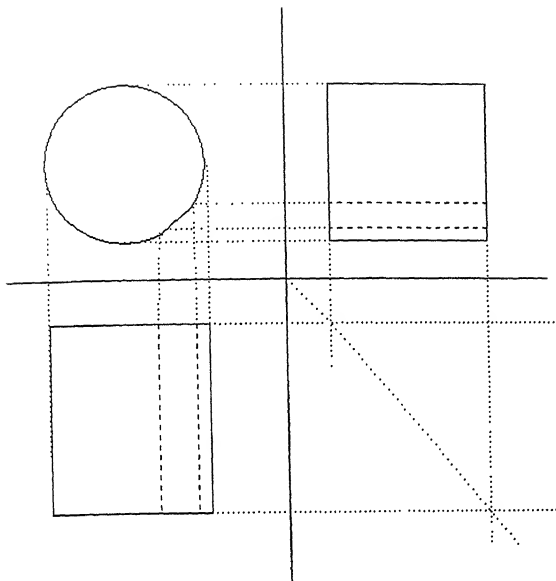


(b) triangle in top view

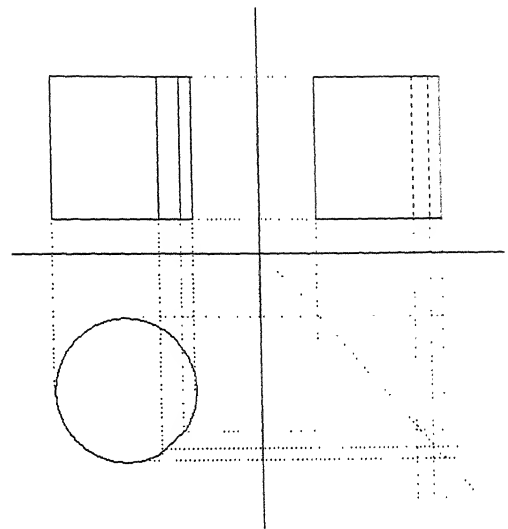


(c) triangle in side view

Fig. 4.17 orthographic views of a triangle in various positions



(a) arc in front view



(b) arc in top view

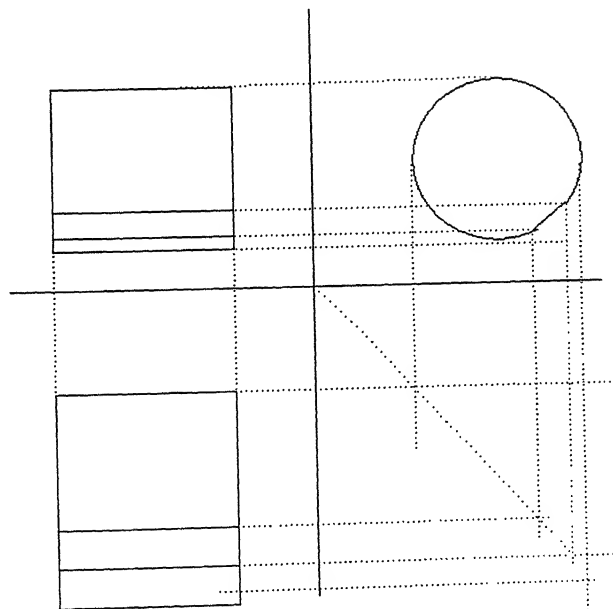


Fig. 4.18 orthographic views of a arc in various positions

4.7 Finding the coordinates of a 3D primitive

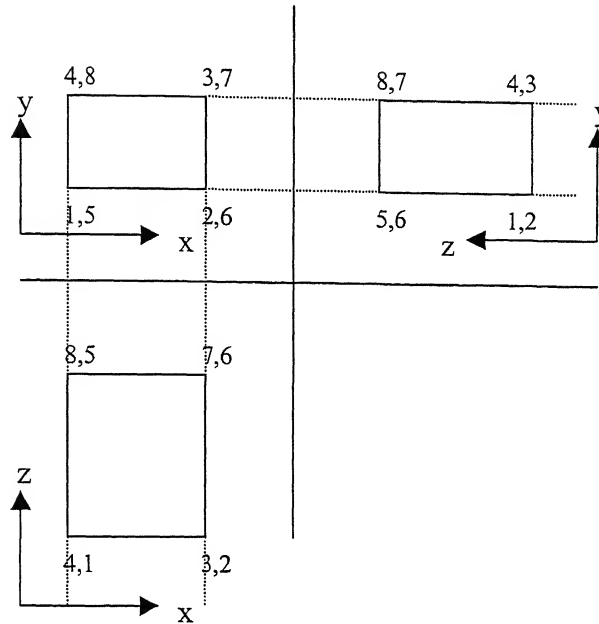


Fig. 4.19: Coordinate System

Figure 4.19 shows the coordinate system used in present problem. The front view has x,y coordinates, top view has x,z coordinates, the side view has y,z coordinates. This coordinate system maps to a 3D object, which is in second octant. The direction of x, y, z axes on the solid are shown in figure 4.20

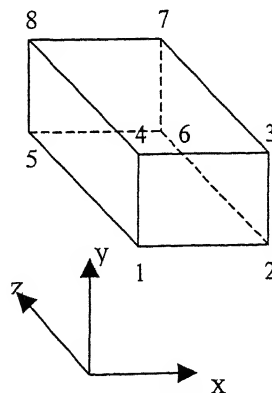


Fig. 4.20: position of the $x-y-z$ axes

Quadrilateral primitive in front view as shown in figure 4.16 matches with some other rectangles in top and side view. Its coordinates in three dimensions can be obtained using any of the two views.

x and y coordinates of point 1 to 8 (figure no.4.16) are same as the x-y coordinates in front view. Their z coordinates can be obtained either from top or side view. If we take top view ordinate of the points gives the z-coordinate. Thus, corresponding x,y,z coordinates of points 1 to 8 can be taken from any of these views.

4.8 Special case of an arc being tangent

In most of the engineering drawings the arc segments are generally tangential to the edges they join. In that case the arc will meet the edge smoothly and hence in the other views it won't form an edge corresponding to it cannot be found. This case has been depicted in figure 4.21

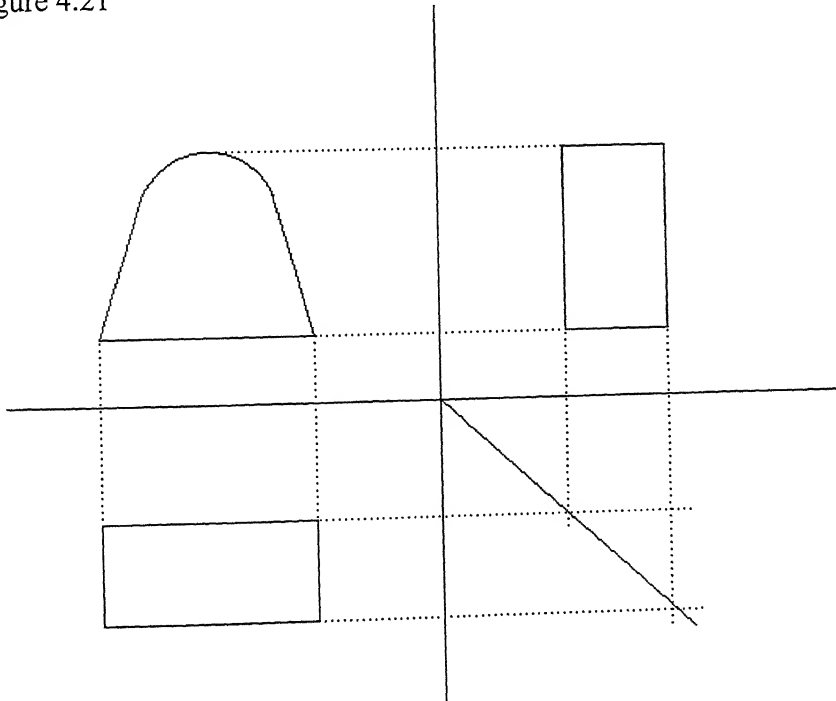


Fig. 4.21 Case of an arc being tangential to an edge

For the view shown, if we separate the arc from the drawing as explained in section 4.5.2 we will get a circular primitive 1-5. While constructing 3D primitives using these 2D primitives, 1-5 cannot find any corresponding 2D primitive in side and top views and the desired result will not be obtained.

In order to solve this problem some connectivity has to be introduced in the other view information. It requires finding those points in the other views, which represent the starting and ending points of this arc segment.

The abscissa and the ordinate range of the polyarc under consideration are noted. Now this polyarc is considered as a rectangle having abscissa and ordinate same as that of this polyarc. Usual discretization procedures are followed and 2D primitives are obtained. For the rectangle considered its corresponding rectangles are found in the other two views. Once these rectangles are found arc segment is projected on to these rectangles and wherever this arc meets them new set of nodes are created and their connectivity is specified between these points. All the algorithms starting from building the tree structure are repeated on the new data. This time for an arc the corresponding edges in the other views are found helping in interpreting them correctly to form the correct solid.

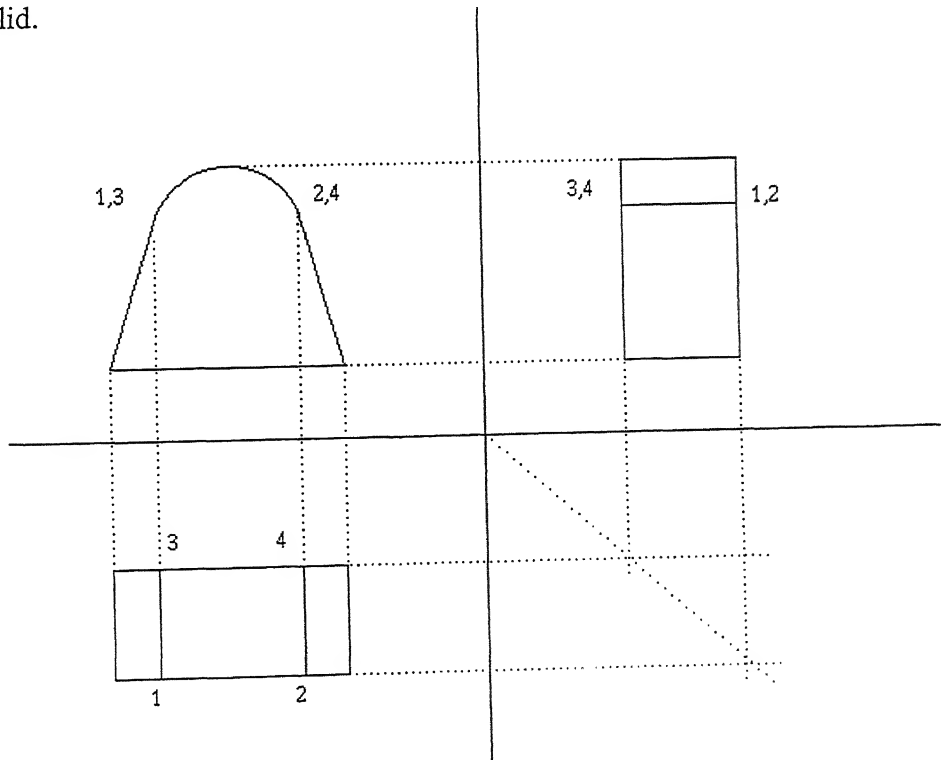


Fig. 4.22 Line Segments introduced in figure 4.21

4.9 Classification of 3D primitives

After forming the 3D primitives from 2D primitives a decision has to be made about the 3D primitive thus formed. 3D primitive formed may be a solid or void. If it is a void it has to be subtracted from the final solid object formed. Hidden line information in the view, sign associated with its corresponding 2D primitives, location in space decides the status of a 3D primitive. Here after 3D primitives are referred as *subparts*.

4.9.1 Sign associated with 2D primitive

If any one of the 2D primitive in front or top or side views corresponding to a subpart is having a negative sign associated with it, then that subpart is a void. If it has positive sign its status can be decided using other information.

4.9.2 Hidden line information

If all of the 2D primitives are having no hidden lines and are joined only by thick lines then, that corresponding subpart is a solid one. If any view is having a hidden line, its status can be decided by its relative position in space. Status of that subpart can't be decided by using the hidden line information.

4.9.3 Position in space

This is the main criterion, which decides the status of a subpart. Here the volume enclosure relations of the different subparts are compared.

Volume enclosure relationships

The volume enclosures relationships between any two subparts will be one of the four cases: IN, WITHIN, ON and OUTSIDE.

- a) A subpart is said to be **IN** another subpart if it is contained by another subpart and shares one or more planes with that subpart.
- b) A subpart is said to be **WITHIN** another subpart if it is within another subpart and shares no plane with that subpart.
- c) A subpart is said to be **ON** another subpart if it is on another subpart and shares one plane with that subpart.
- d) A subpart is said to be **OUTSIDE** another subpart if it is outside another subpart and share no plane with that subpart.

Rules to classify subparts

- In another subpart, the outer one is solid and the inner one is void.
- If a subpart is **IN** in two of the subparts, then the outermost and the inner most subparts are solids. The intermediate one is void. This case is shown in fig. 4.23

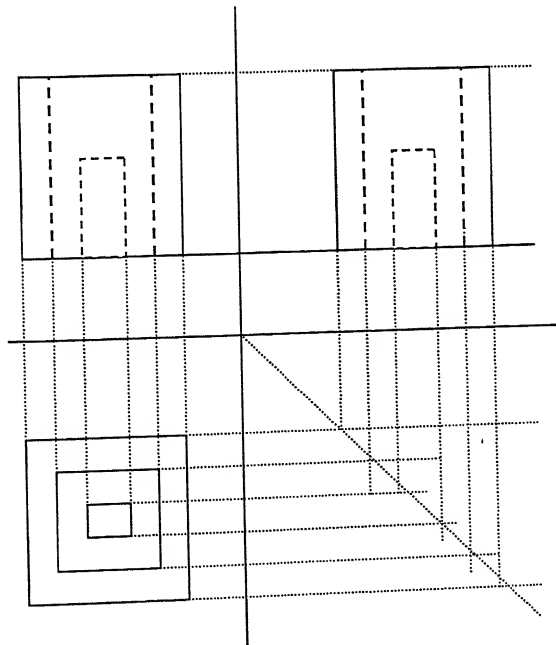


Fig.4.23: Case of one subpart is IN in two other subparts

The algorithm described above in section 4.4 to 4.6 classifies the view shown in figure 4.23 as three subparts. As there are 2 subparts which are IN in the outermost subpart, the inside one and outermost one are only solid and the intermediate one is void.

- If a subpart is IN If a subpart is OUTSIDE other subpart, then they are left unclassified.
- If a subpart is ON any other subpart, then they are not classified

Intersecting subparts

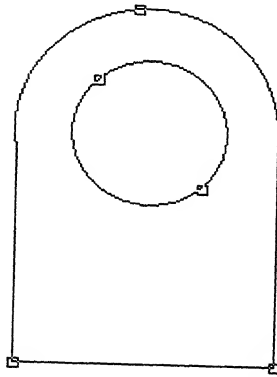
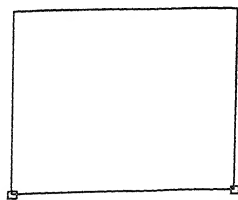
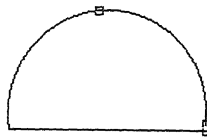


Fig. 4.24 View of a typical orthographic drawing

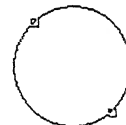
View of a drawing as shown in the figure 4.24, after constructing subparts results in forming a cylinder, a rectangular prism and another half cylinder as shown in figure 4.25.



(a) Subpart 1



(b) Subpart 2



(c) Subpart 3

Fig. 4.25: Subparts after reconstruction for the figure 4.24

Subpart 3 intersects partially with subpart 1 and subpart 2. This case occurred as the loops has been discretized to form 2-D primitives. Subpart 1 and subpart 2 are formed by discretization of the outer loop of the drawing shown in the figure 4.24. Half of the subpart 3 is in subpart 2 and other half is in subpart 1. As a whole subpart 3 is completely IN inside the subpart joined by subpart 1 and subpart 2. Thus, if a subpart is partially IN or WITHIN in any one of the subparts and the other part of it is IN or WITHIN in any other subpart, the subpart under consideration is IN and so is void.

All the other cases of intersection are left unclassified.

After the completion of classification of subparts, if some subpart is left unclassified it means that it does not have any relation with other subparts. If its 2D primitives are having hidden lines, it is only due to loss of semantics. So, those parts left unclassified are solid.

4.10 Boolean operations

Objective of this step is to do Boolean operations on the 3D subparts obtained. Each 3D subpart is modeled using I-DEAS and using the boolean operation option available in the package, the final 3D object is realized.

पुस्तकालय संशोधनार्थ केवलक पुस्तकालय
सं. 141967
अवधि क. A 141967

Chapter 5

Results

The algorithms described in Chapter 4 are implemented in case of some practical engineering drawings and correct results are obtained. Implementation is done using C language on an IBM computer with Pentium *III* processor under Red hat Linux 7.0. Boolean operations are performed using I-DEAS Master Series software on a COMPAQ computer under Windows NT 4.0. Results obtained are shown in the figure 5.1 to 5.11. Graphics used in the results are drawn using graphics library OpenGL.

The first three cases shown in figure 5.1,5.2,5.3 represent the basic steps in the reconstruction process. This three cases form the knowledge base of the thesis. Any other subsequent case can be thought of involving these three things only along with some regularized boolean operation. These three parts does not require any boolean operations. The output is shown using OpenGL implementation. I-DEAS is not required for dealing these cases (rectangular prism, triangular prism, cylinder). Other figures 5.4-5.11 show same of the practical engineering drawings. Reconstruction involves forming subparts, these subparts has to be joined using regularized Boolean operations.

Figure 5.4 describe a drawing where rectangular and triangular primitives are only involved. This object is one of the simplest objects that can be formed using triangular and rectangular primitives. Discretization of these views resulted in forming a set of rectangles and triangles and after corresponding matching 3D primitives are constructed.

Figure 5.5 shows a case where only rectangular primitives are involved. Ray casting method completely discretized the views. As the view contains some hidden lines volume enclosure relations are necessary to classify the subparts. After classifying they are processed in I-DEAS for doing Boolean operations.

Figure 5.6 and 5.7 present drawing having circular and rectangular primitives. Ray casting is involved while forming 2D-primitives. The case of an arc being tangent to the edge occurred. 3D primitives are classified using volume enclosure properties.

Figure 5.8 shows a case involving circular, rectangular, triangular primitives. All triangular primitives formed are positive. Ray casting is used to separate rectangular

primitives. Some arcs in the drawings are tangential and some are not. Volume enclosure relations are also needed.

Figure 5.9 shows a case where triangular, circular, rectangular primitives are involved. The triangular primitives are separated directly. Ray casting, volume enclosure relations are also needed.

Figure 5.10 shows a case involving circular, rectangular, triangular primitives. There the 2D-triangular primitives formed are negative. Ray casting, volume enclosures are also used.

Figure 5.11 shows a practical engineering drawing with rectangle, triangle, and cylindrical primitives. Arcs are tangential to some edges while some edges are not tangential to one of the arc. Ray casting, inclined line separations are used to form 3D subparts. Volume enclosure relations classified the subparts. Boolean expressions are performed using I-DEAS.

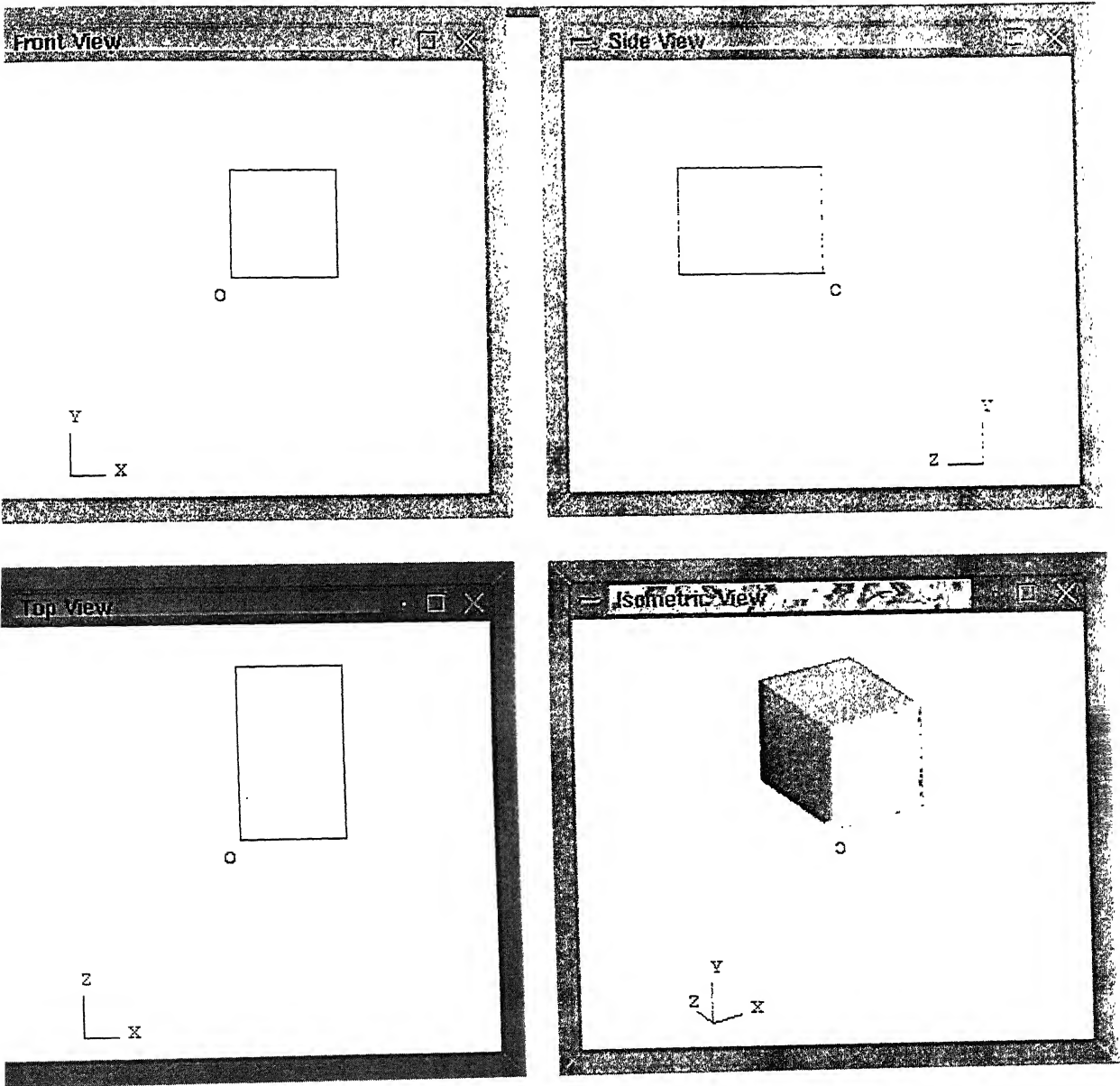


Fig 5.1: Reconstruction of a rectangular prism.

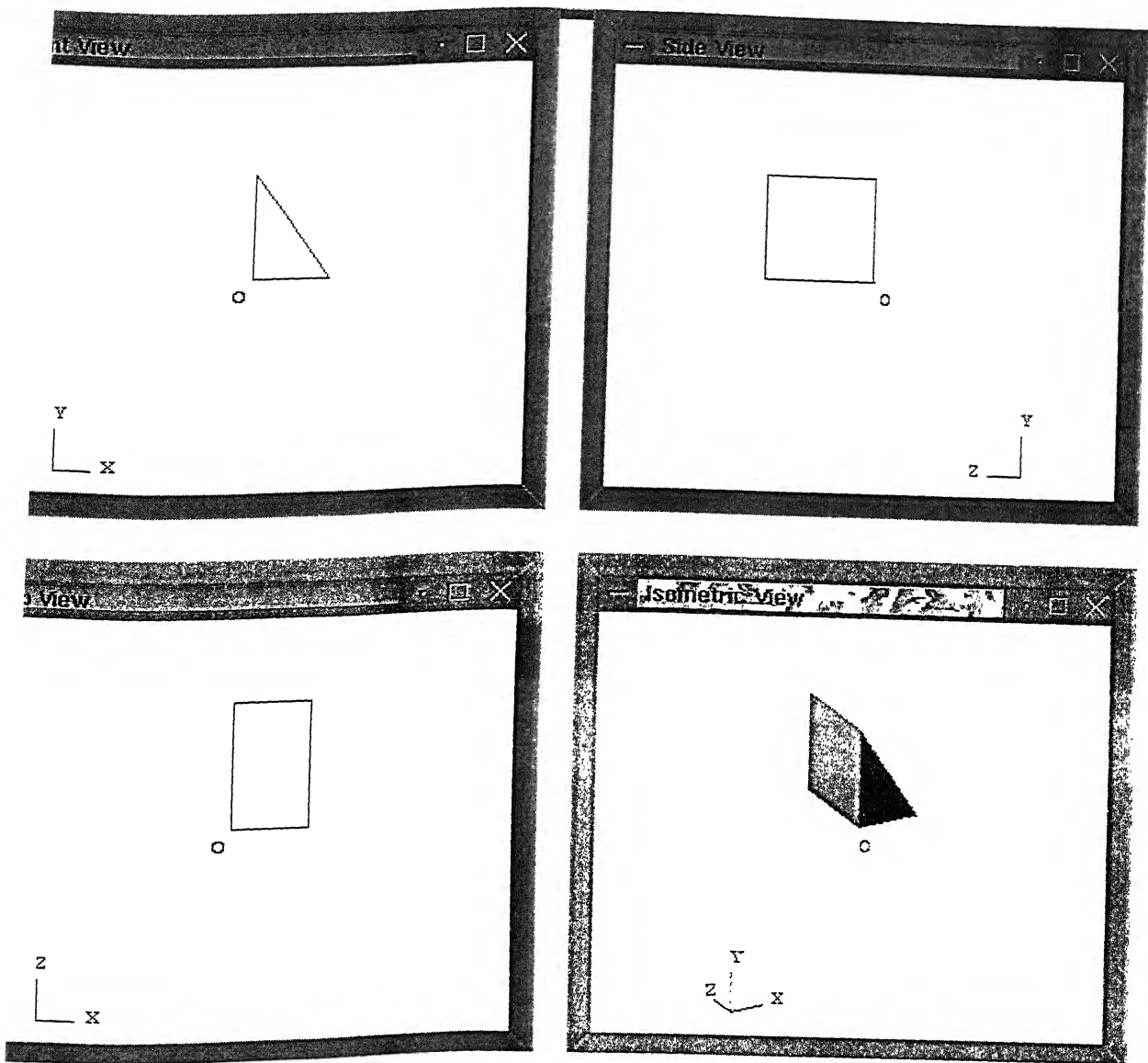


Fig 5.2: Reconstruction of a triangular prism.

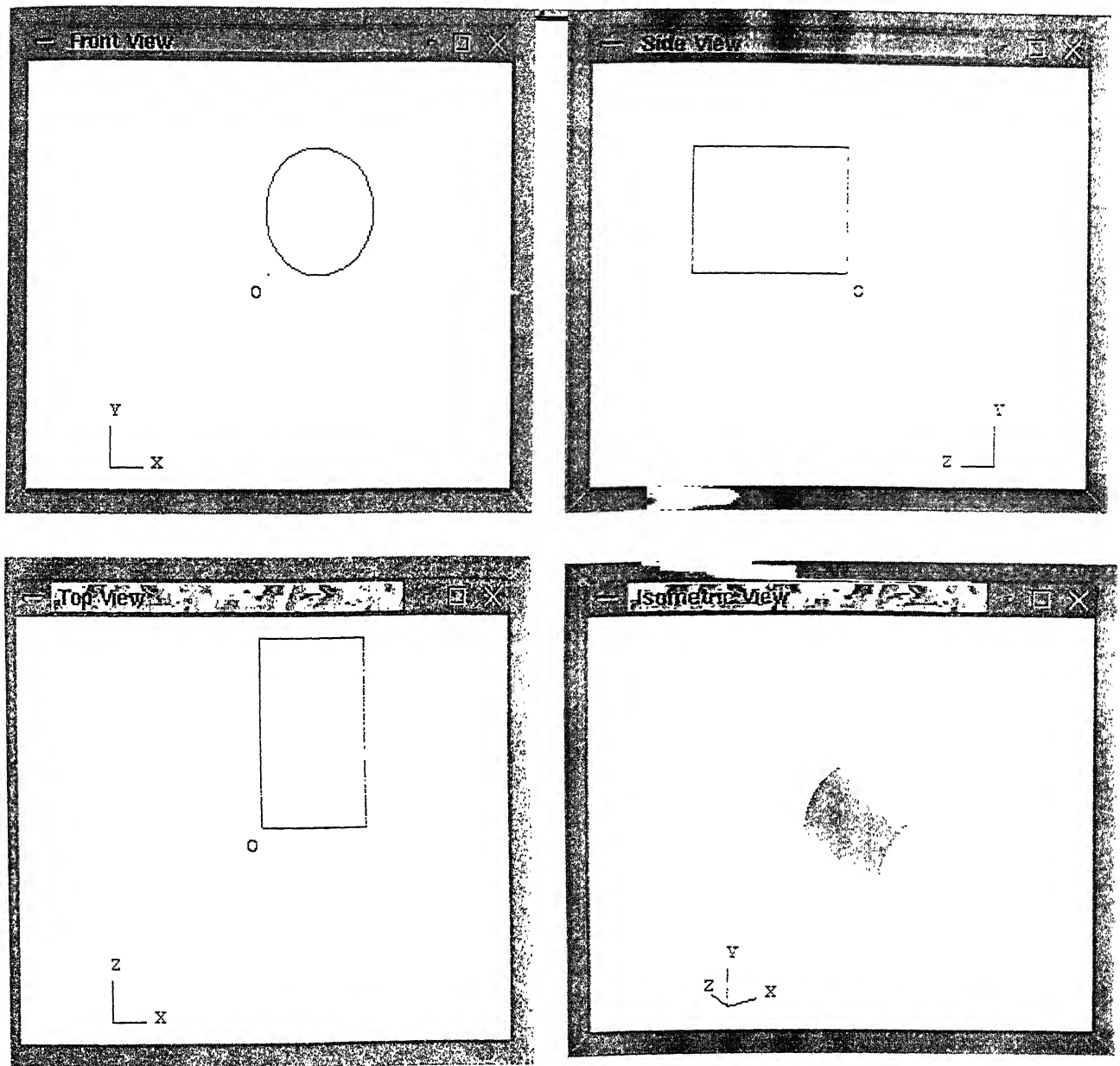


Fig 5.3: Reconstruction of a cylinder.

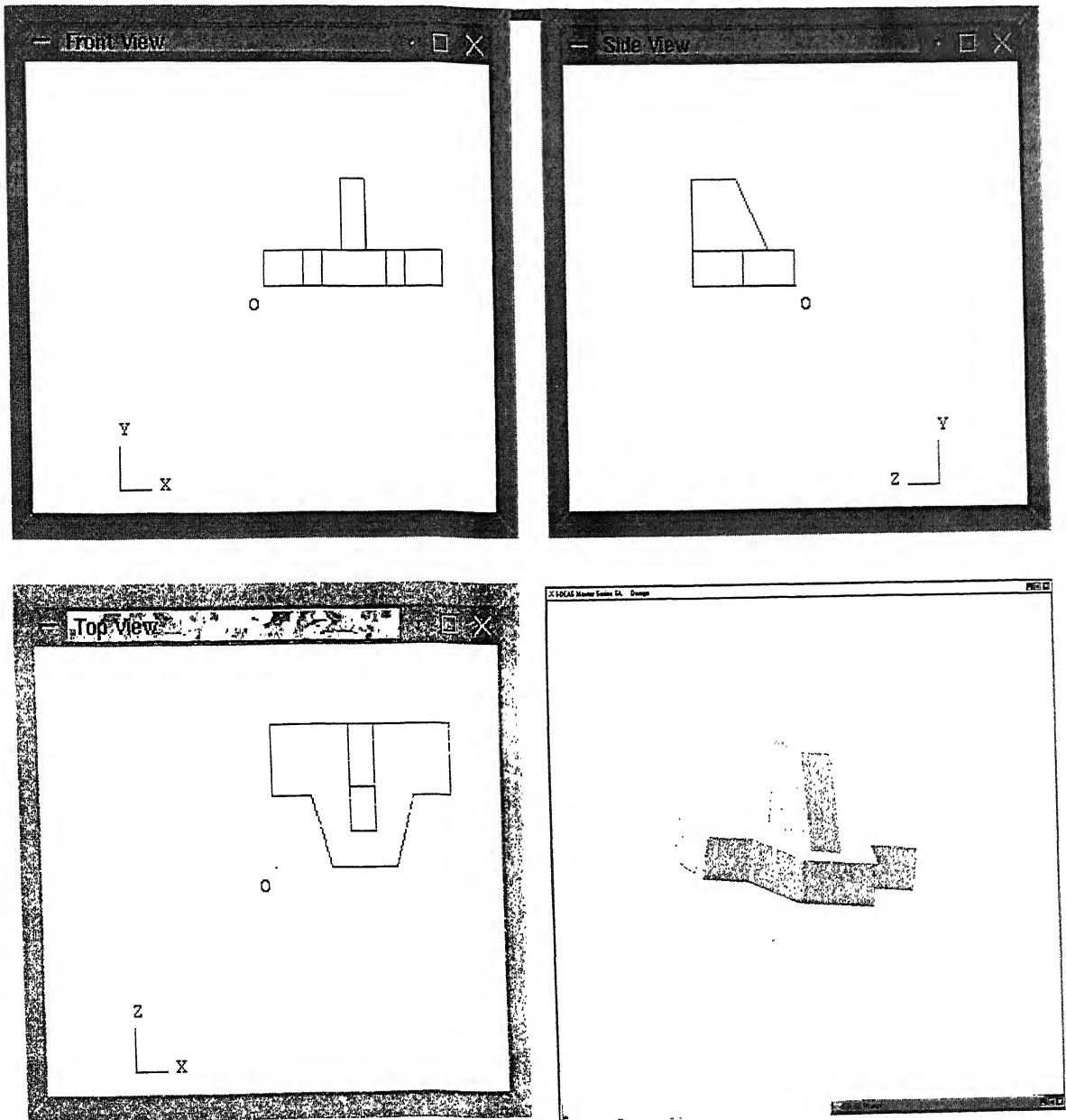


Fig 5.4: Reconstruction of an object having rectangular, triangular primitives.

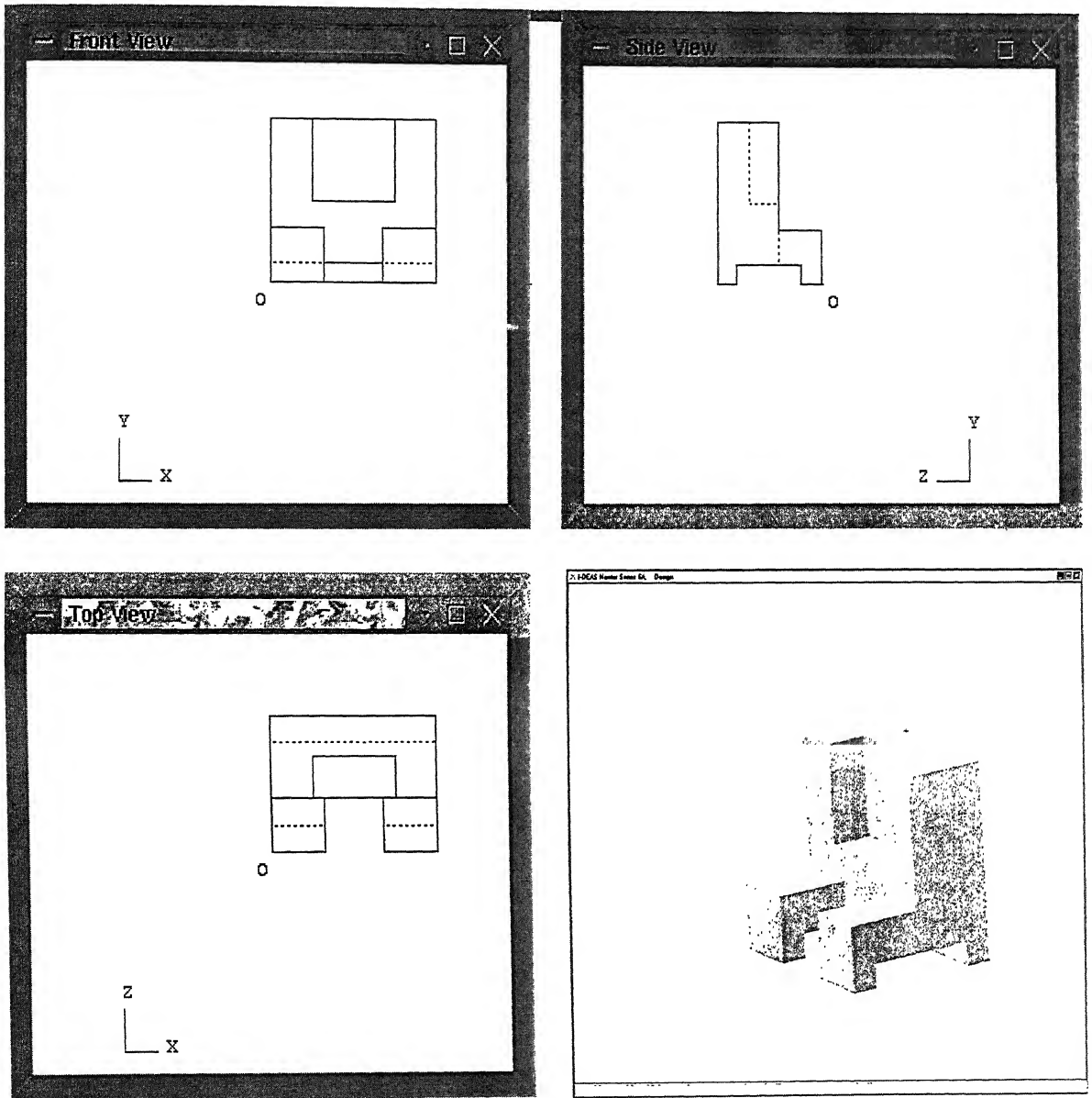


Fig 5.5: Reconstruction of an object having rectangular primitives.

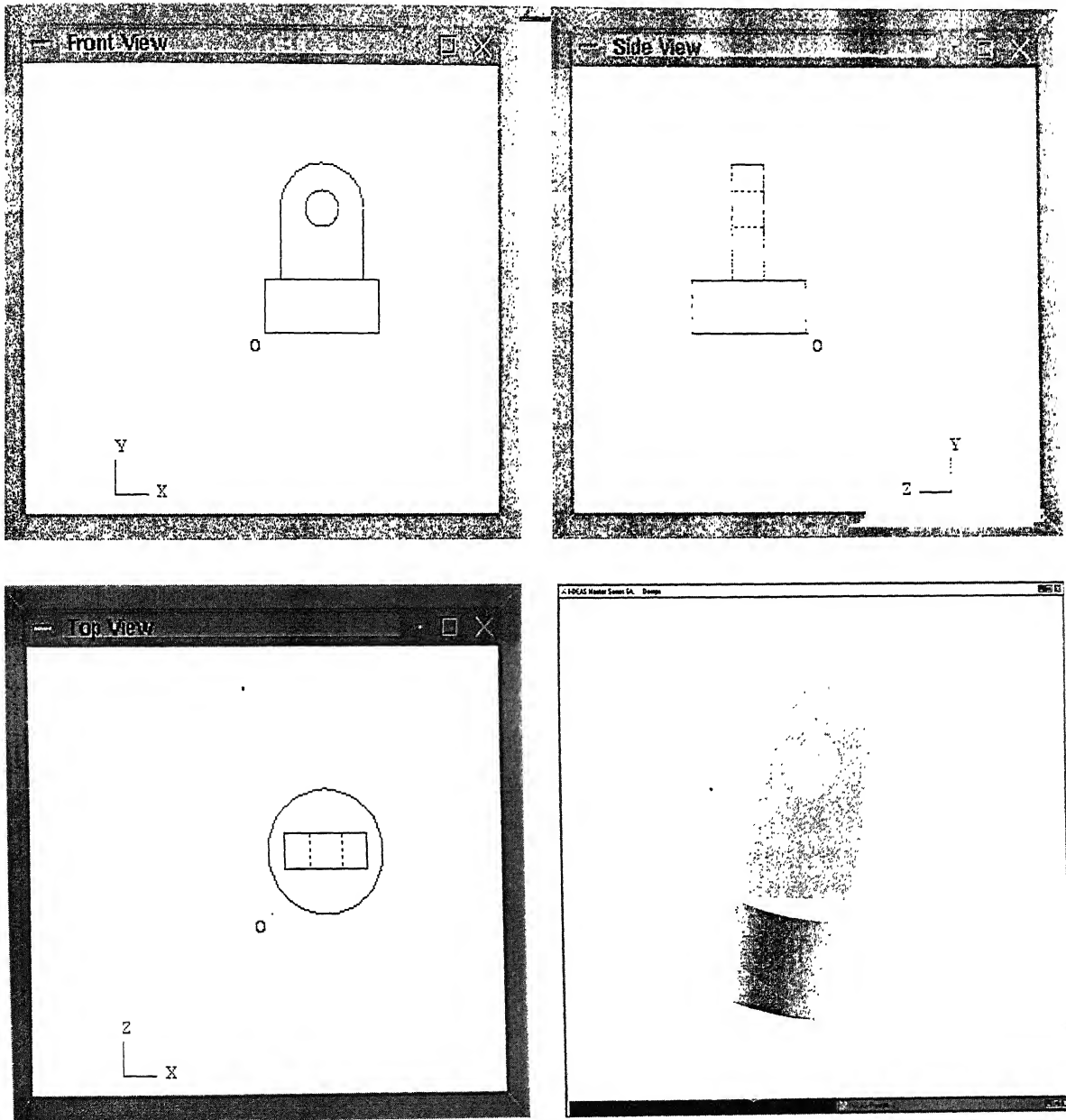


Fig 5.6: Reconstruction of an object having rectangular, circular primitives case- (i)

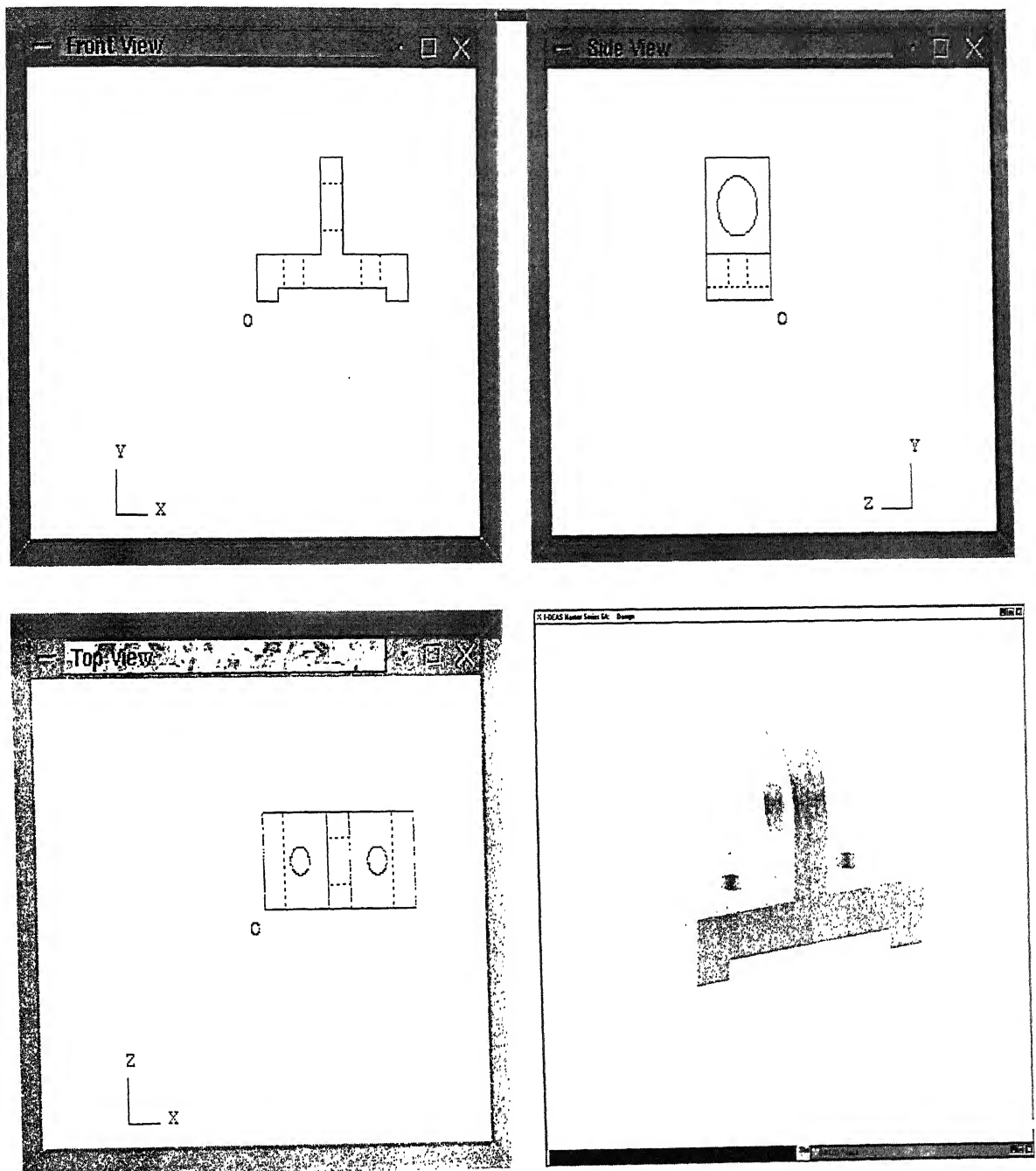


Fig 5.7: Reconstruction of an object with rectangular, cylindrical primitives case- (ii)

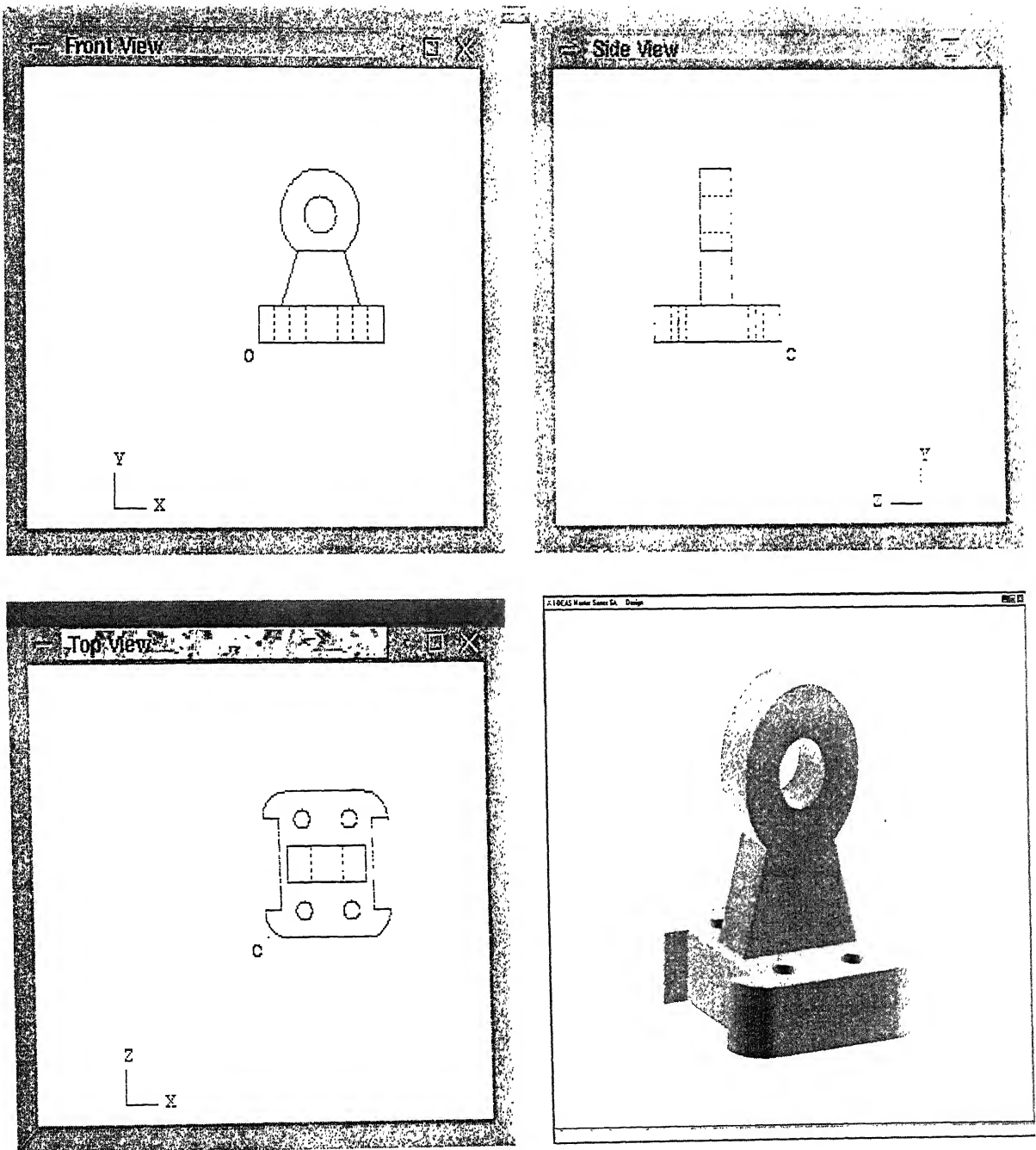


Fig 5.8: Reconstruction of an object with rectangular, triangular, cylindrical primitives case- (i)

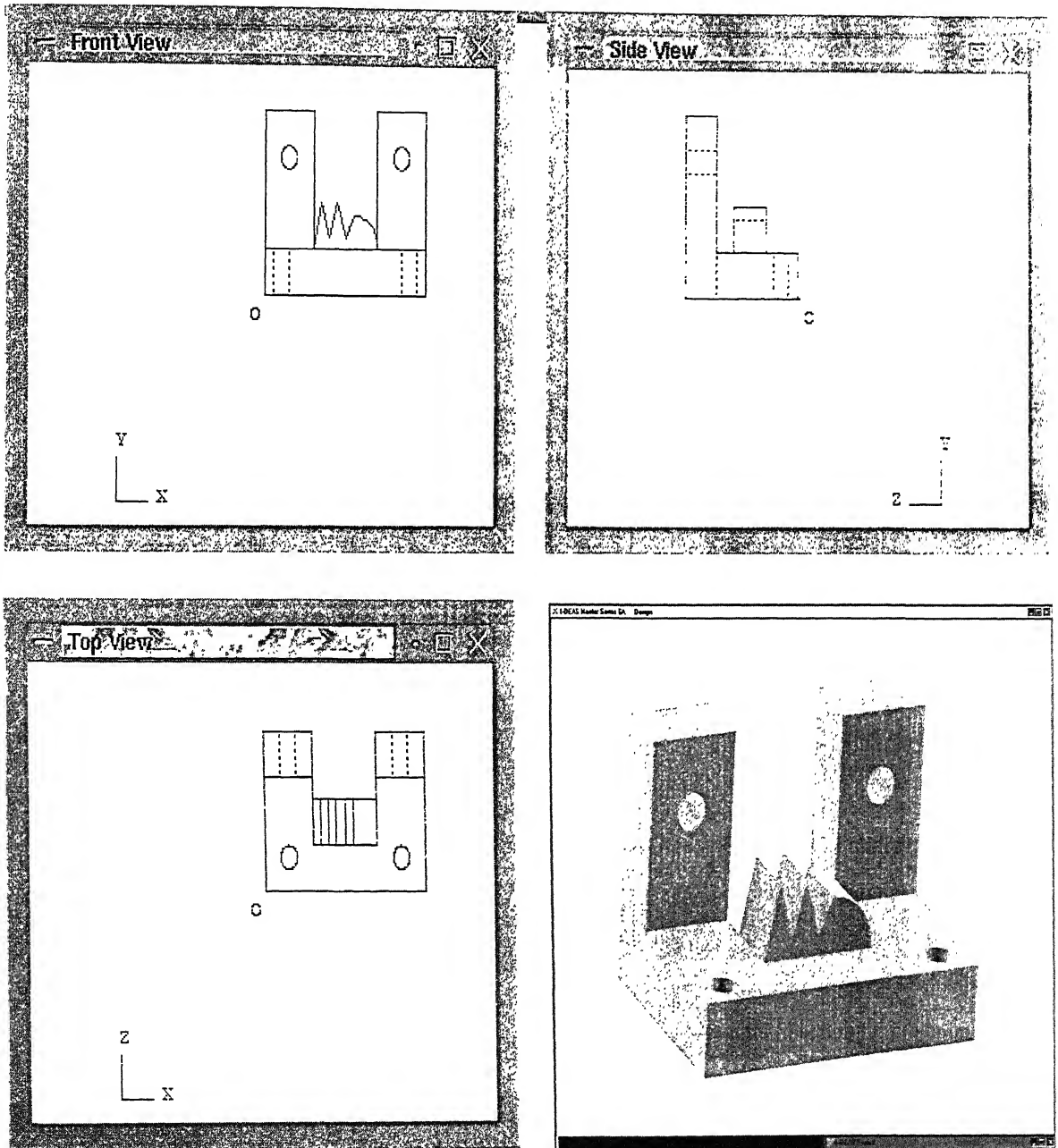


Fig 5.9: Reconstruction of an object with rectangular, triangular, cylindrical primitives case- (ii)

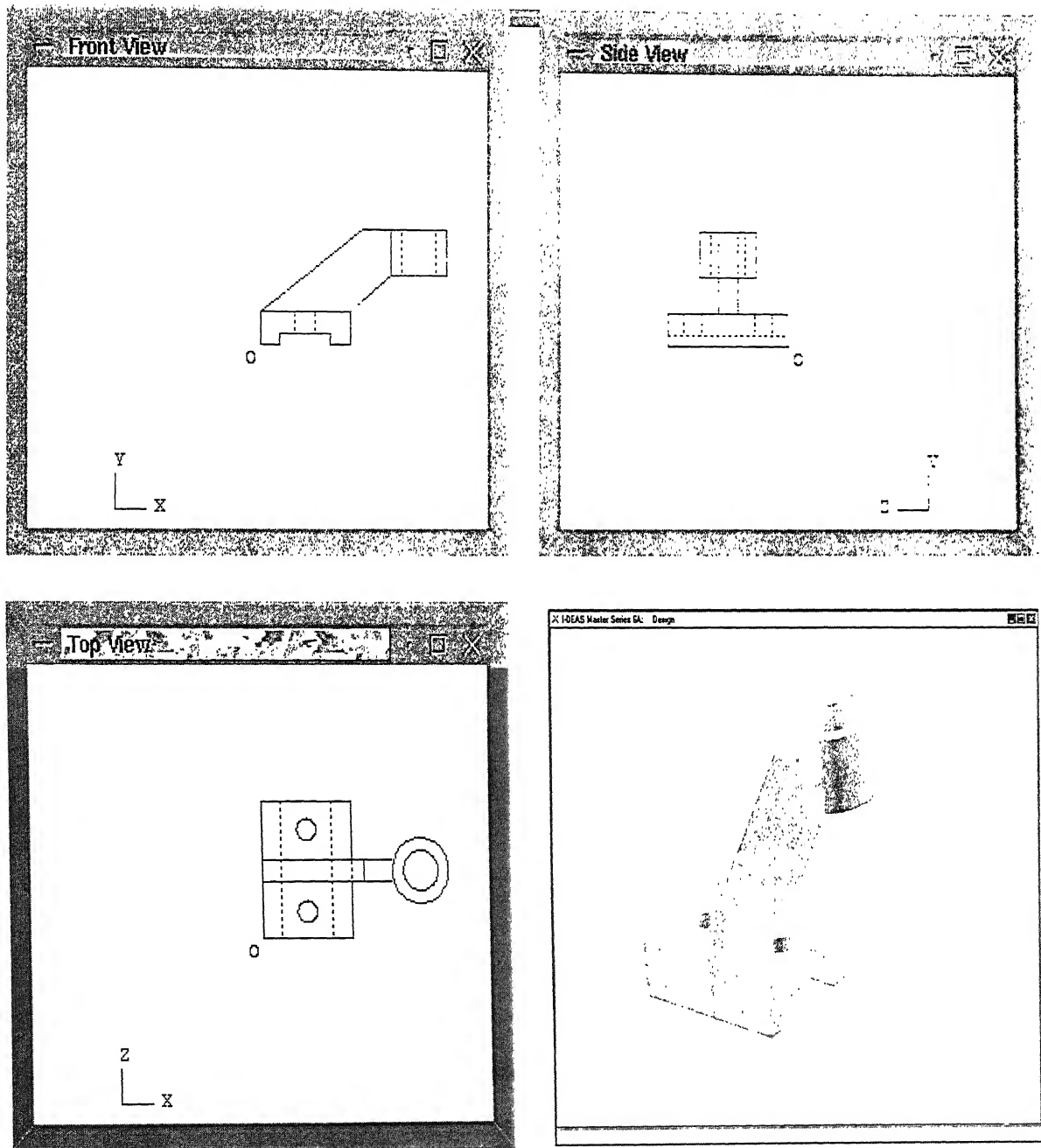
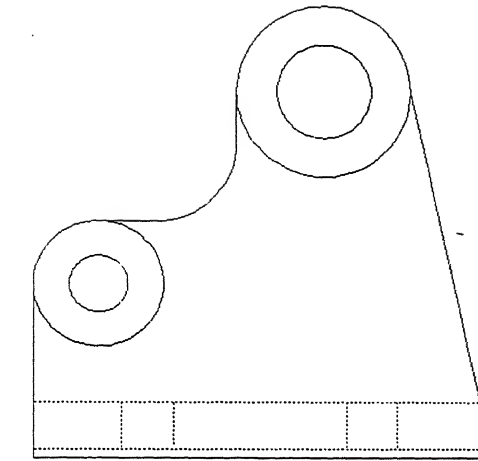
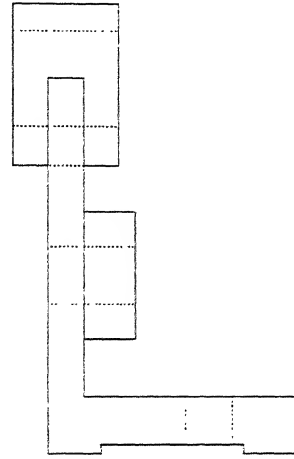


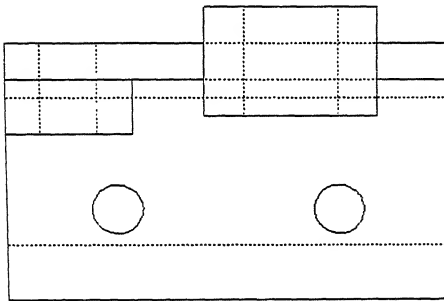
Fig 5.10: Reconstruction of an object with rectangular, triangular, cylindrical primitives case- (iii)



o



c



o

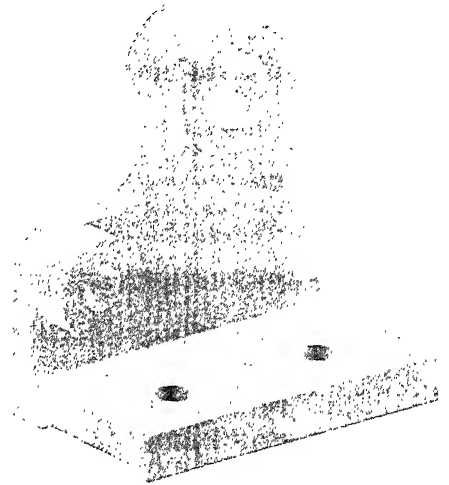


Fig 5.11: Reconstruction of an object with rectangular, triangular, cylindrical primitives case- (iv)

Chapter 6

Conclusions and Scope for Future Work

An algorithm has been developed and implemented to generate a solid model from three orthographic views namely front, top & side view. Polyhedral objects as well as objects having cylindrical edges have been handled successfully. The work presented is computationally efficient and very fast. As the views are treated as set of primitives each edge and point need not be considered separately reducing the number of comparisons to be made.

6.1 Achievements

1. The application of reconstruction algorithm to some practical drawings has shown that automatic reconstruction can be achieved.
2. Hidden line information of views is successfully handled.
3. Algorithm resulted in 3D CSG objects and their edges, planes, instead of 3D vertices, edges, planes which are hard to decide whether they are spurious or true.
4. Resultant solid model obtained is complete and can be directly given as input to various procedures like FEM, NC machining.
5. This algorithm does not involve sweeping operations, which are hard to implement, time consuming. Therefore, it reduced processing time compared to other works done using volume-oriented approach.
6. Some types of wrong input can be handled if it occurs only in one view and correspondence for this error is not found in another view.

6.2 Limitations

1. Input has to be complete. The lines, which can be ignored by a trained person while drawing projections, cannot be ignored while giving input to software developed here. No tolerance is given for human errors.

2. Although it can deal with cylindrical, polyhedral objects; Other objects with conical, spherical geometrics cannot be dealt.
3. It cannot understand some special engineering symbols such as threads, fillet etc.
4. Only three projections can be used. Sectional view or auxiliary view cannot be used to provide some extra information which otherwise is very difficult to represent in engineering drawing.
5. It requires three views for reconstruction even only two views are necessary in some cases.

In spite of these limitation the work presented here shows a direction of a new automated approach to generate 3D objects. Continuation of this work may lead to a completely automatic solid modeler, which is a handy tool for any design engineer.

6.3 Scope for future work

1. This work is limited to cylindrical, polyhedral surfaces. So this can be extended for conical and spherical surfaces.
2. Since humans prepare engineering drawings, human error should be tolerated. The input may not be exact, *i.e.*, some connectivity might be missing in input or some wrong connectivity might be there. The work dealt here requires exact input and may generate some irrelevant object if input itself is not complete. It is not possible to take into consideration all the types of human errors but some very common mistakes can be taken care for.
3. An algorithm to perform the regularized 3D Boolean operations can be made to get result without using I-DEAS package.
4. Final aim of any software dealing with solid model reconstruction using orthographic projection is to take scanned image of a blue print drawing as input. An image processing software can be made which can take the scanned blue print, separate layer of geometry and gives it as input to this program.

5. This algorithm cannot identify mechanical engineering symbols such as fillet, threads etc. So an algorithm can be added to this can identify these special symbols in three views and generate it in final three-dimensional object.
6. Some engineering drawing can be interpreted from two views only. So this feature also can be added to present system. A system than can be established which will generate 3D object from two views if unique solutions exists, otherwise it should generate all possible solutions.
7. Sectional and auxiliary views are used to represent some special features of object. If an algorithm can be developed to utilize their information then some other engineering drawings can be solved.
8. In present algorithm knowledge of quadrilateral, circular arcs, and a triangle is used. We can widen this knowledge base to include some more complicated shapes.
9. Knowledge base can be further extended to make the computer to understand the symbols used in an engineering drawing.

Bibliography

- [1] Idesawa, M (1973). A system to generate a solid figure from three views. *Bull Japanese society of mechanical Engineering* Vol. 16, pp 216-225.
- [2] Idesawa, M. and shibata, S. (1975). Automatic input of Line Drawing and Generation of Solid Figure from Three View Data. *Proc. Int. Comp. Sumposium* Vol 2, pp 216-225.
- [3] Shapira R.(1974). A Technique for the Reconstruction of straight-Edge, Wireframe Object from Two or More Central projections. *Computer Graphics and Image Processing* Vol. 3, pp 318-326.
- [4] Markowasky, G. and Weseley M.A. (1980). Fleshing Out wireframes: Reconstruction of Objects, *part ii IBM J. Research and Dev.* Vol. 25, No. 6, pp 934-954.
- [5] Markowasky, G. and Weseley M.A. (1980). Fleshing Out wireframes: Reconstruction of Objects, *part i IBM J. Research and Dev.* Vol. 24, No. 5, pp 582-597.
- [6] Sakurai, H. (1983). Solid Model Input Through Orthographic Views. *Computer Graphics*, Vol. 17, No. 3 pp 243-247.
- [7] Yan, Q., Chen, C. L. P. and Tang, Z. (1994). Efficient Algorithm for the Reconstruction of 3D Objects from Orthographic Projections. *Computer-Aided-Design* Vol. 26, No. 9, pp699-717.
- [8] Mukarjee, A., Sasmal, N. and Sastry, D. S. (1999). Efficient Categorization of 3D Edges from 2D Projections. *GREC'99, LNCS* pp 288-297.
- [9] Press, K. (1984). Constructing The Solid representation From Engineering projections. *Computer Graphics* Vol. 8, No. 4, pp 381-389.

- [10] Gujar, U. and Nagendra, I. (1989). Construction of 3D Objects from Orthographic Views. *Computer Graphics* Vol. 13, No. 4, pp 505-521.
- [11] Shin, B. S. and Shin Y. G. (1994). Fast 3D Model Reconstruction from Orthographic Views. . *Computer-Aided-Design*.
- [12] Tombre, K. and Ah-Soon, C. (1995). A step Towards Reconstruction of 3D CAD Models from Engineering Drawings. ICDAR'95.
- [13] Shpitalni, M. and Lipson, H. (1996). identification of Faces in 2D Line Drawing Projection of a Wireframe Object. *IEEE Trans. On Pattern Analysis and Machine Intelligence* Vol. 18, No. 10, pp 1000-1012.
- [14] Kuo, M.H. (1997). Reconstruction of Quadratic Surface Solids from Threee-View Engineering Drawings. . *Computer-Aided-Design* Vol. 30, No. 7, pp 517-527.
- [15] Masuda, H. and Numao, M. (1997). A Cell-Based Approach for Generating Solid Objects from Orthographic Projections. *Computer-Aided-Design* Vol. 29, No. 3, pp177-187.
- [16] Dori, D. (1989). A Syntatic /Geometric approach to Dimensions in Engineering Machine Drawings. *Computer Vision, Graphics and Image Processing* Vol. 47, No. 3, pp 271-291.
- [17] Tombre, K. and Dori, D. (1995). From Engineering Drawings to 3D CAD Models: Are we Ready Now? *Computer-Aided-Design* Vol. 27, No. 4, pp243-255.
- [18] Haralick R.M. and Shaprio. (1982). Understanding Engineering Drawing. *Computer Graphics and Image Processing* Vol. 20, pp 244-258.

- [19] Aldefeld, B. (1983). On Automatic Recognition of 3D structures from 2D representation. *Computer-Aided-Design* Vol. 15, No. 2, pp 59-63.
- [20] Aldefeld, B. and Richter, H. (1984). Semianutomatic Three-Dimensional Interpretation of Line Drawing. *Computer Graphics* Vol. 8, No. 4, pp 371-380.
- [21] Bin, H. (1986). Inputting Constructive Geometry Reoresentations Directly from Orthographic Views. *Computer-Aided-Design* Vol. 18, No. 3, pp147-155.
- [22] Chen, Z and Perng, D. B. (1988). Automatic Recognition of 3D Solid Objects from 2D Orthographic Views. *Pattern Recognition* Vol. 21, No. 5, pp 439-449.
- [23] Meeran, S. and Pratt M. J. (1993). Automatic Feature Recognition from 2D Drawings. . *Computer-Aided-Design* Vol. 22, No. 5, pp7-17.
- [24] Tanaka, M., Iwama, K., Hosoda, A. and Watanabe, T. (1998). Decomposition of a 2D Assembly Drawing into a 3D part Drawings. *Computer-Aided-Design* Vol. 30, No. 1, pp37-46.
- [25] Shum, S. S. P., Lau, W.S., Yuen, M. M. F. and Yu, K.M. (2000). Solid Reconstruction from Orthographic Views using 2-Stage Extrusion. *Computer-Aided-Design* Vol. 33, pp91-102.
- [26] Tomiyama, K. and Nakaniwa, K. *Reconstruction of a 3D solid Model from Three Orthographic Views*. TOP-DOWN Approach.
- [27] Ibrahim Zeid, *CAD/CAM Theory and Practice*, Tata McGraw-Hill edition
- [28] Tenenbaum, M. Aaron, Yedidyah Langsam, Augestein, J. Moshe, *Data structures using C*, Prentice-Hall of India Private Limited.
- [29] Sidheswar, Kannaiah , *Machine Drawing*, Tata McGraw-Hill edition.

Appendix

Input Format

In this thesis input to the computer program developed, is given manually. Input is prepared by observing the orthographic view. Corresponding to the three orthographic views three input files have to be generated and subsequently fed as input.

Input format is designed in such a way to consider all the geometric details are covered. Certain conventions are followed to provide ease in giving input. User has to be careful to give the correct input, as some mistake in connectivity may lead to some irrelevant object.

Input is basically in the form of connectivities, and the way the nodes are connected. Design of the input one can say is node centered. The type of connection, whether is by dashed line which represents a hidden line or a thick line has also to be mentioned.

For each node its coordinates in the two dimensional space are to be specified along with the number of its neighboring nodes. Then for each neighboring node, the way it is connected to the present node with the details of the connectivity is to be specified.

Notations used

Some notations are followed in the input. These notations are basically in binary format and used to differentiate between to things, which are quite similar to each other. They are given as follows

- Line connectivity 0, arc connectivity 1
- Dashed line 0, thick line 1
- CW direction 0, ACW direction 1

Points connected by a line segment

If two nodes a and b are connected by a line segment, input for the node a contains its coordinates, number of neighboring nodes to a and the type of connection whether it is dashed or thick.

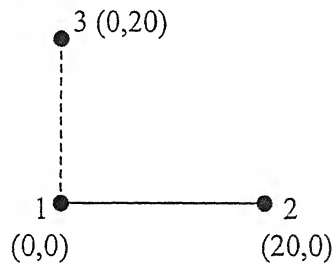


Fig. A-1: nodes connected by line segments

For the figure A-1 the input is as follows

1 0 0 2

2 0 1

3 0 0

where,

1 - node number

0 - its abscissa

0 - its ordinate

2 - number of neighboring nodes

2 - neighboring node

0 - line connection

1 - thick line

3 - neighboring node

0 - line connection

0 - dashed line

Points connected by an arc segment

For an arc segment connecting two points, the input gives the details of the arc. There are many ways of specifying the arc. In this thesis its center coordinates, radius, starting angle, ending angle, sense represent an arc segment. The starting and ending angles of the arc are measured in CW direction.

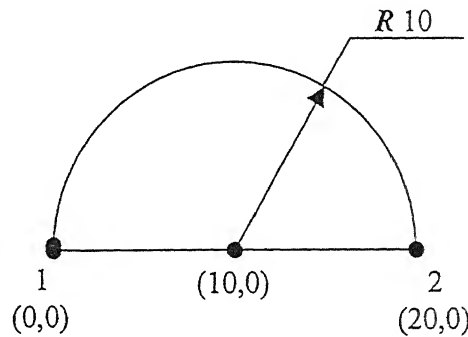


Fig. A-2: nodes connected by an arc segment

For the figure A-2 the input is as follows.

1 0 0 1

2 1 10 0 10 180 360 0 1

where,

1 - node number

0 - its abscissa

0 - its ordinate

1 - number of the neighboring nodes

2 - neighboring node

1-arc connection

10- abscissa of the center of the arc

0 - ordinate of the center of the arc

10 - radius of the arc

180 - starting angle of the arc

360 - starting angle of the arc

0 - CW arc

1 - thick line

This way the input is specified for each node in the drawing.

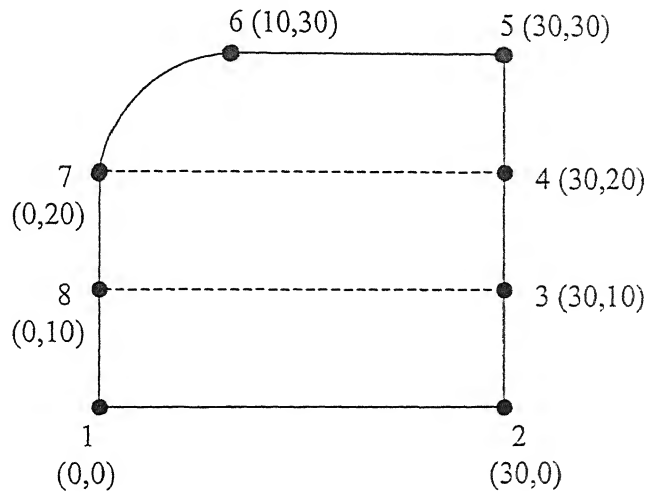


Fig. A-3: A typical orthographic view

For the drawing shown in figure A-3 the input is specified as follows.

```

1 0 0 2
2 0 1
8 0 1
2 30 0 2
1 0 1
3 0 1
3 30 10 3
2 0 1
4 0 1
8 0 0
4 30 20 3
3 0 1
5 0 1
7 0 0
5 30 30 2
4 0 1

```

6 0 2

6 10 30 2

5 0 1

7 1 10 20 10 180 270 1 1

7 0 20 3

4 0 0

6 1 10 20 10 180 270 0 1

8 0 1

8 0 10 3

1 0 1

3 0 0

7 0 1

A 141967



A141967